Informatica® Cloud Data Integration

Google BigQuery Connectors

Table of Contents

Preface

Use *Google BigQuery Connectors* to learn about *Google BigQuery* and *Google BigQuery V2 Connectors*. Use Part I to know the overview and the functionality comparison between Google BigQuery V2 and Google BigQuery connectors. Use Part II and Part III to learn about the functionality available for Google BigQuery connectors.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center through the Informatica Network or by telephone.

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

Part I: Introduction to Google BigQuery connectors

This part contains the following chapters:

CHAPTER 1

Google BigQuery connectors overview

You can use Google BigQuery connectors to read data from and write data to Google BigQuery. Use the connectors to create sources and targets that represent records in Google BigQuery.

When you use Google BigQuery connectors to create and run a Data Integration task, the Secure Agent reads from and writes data to Google BigQuery based on the taskflow and Google BigQuery connection configuration.

You can move data from any data source to Google BigQuery. Use the following connectors to create connections and integrate data to and from Google BigQuery:

Google BigQuery V2 Connector

This is the recommended connector to connect to Google BigQuery. Use Google BigQuery V2 Connector to create a mapping, dynamic mapping task, data transfer task, or a mapping task.

Google BigQuery Connector

This is an older version of Google BigQuery Connector. Use Google BigQuery Connector to read or write data only when you want to use a synchronization task.

CHAPTER 2

Connector comparison

Based on your requirements to integrate or ingest data, you can use either Google BigQuery Connector or Google BigQuery V2 Connector to create data integration tasks.

The functionality to create integration tasks and configure read and write operations differ in both connectors. Informatica recommends to use Google BigQuery V2 connector as the new features and enhancements are provided for Google BigQuery V2 Connector.

Mapping functionality

The following table compares the mapping functionality supported by Google BigQuery connectors:

Mapping functionality	Google BigQuery Connector	Google BigQuery V2 Connector
Hosted agent	Yes	Yes
Proxy server	Yes	Yes
Synchronization task	Yes	No
Mapping task	Yes	Yes (Preferred)
Dynamic mapping task	No	Yes
Data transfer task	No	Yes
Write CDC data to Google BigQuery Targets	No	Yes
SQL ELT optimization	Use an ODBC connection with **ODBC Subtype**=`Google BigQuery` to enable source or full SQL ELT optimization between Google BigQuery source and target.	Use a Google Cloud Storage V2 source connection and an Google BigQuery V2 target connection to enable full SQL ELT optimization between Google Cloud Storage source and Google BigQuery target. **Note:** SQL ELT optimization does not apply to mappings in advanced mode.

Mapping functionality	Google BigQuery Connector	Google BigQuery V2 Connector
Lookup transformation	No	Cache, uncached, connected, and unconnected **Note:** Uncached lookups, Unconnected with cached lookups, and dynamic lookup cache do not apply to mappings in advanced mode.
SQL transformation	No	Yes **Note:** SQL transformation does not apply to mappings in advanced mode.

Source functionality

When you import an object from Google BigQuery to read data, you can configure the advance source properties to determine the read operation behavior. For example, you can read data in staging mode or you can configure pre-SQL and post-SQL configurations.

The following table lists the source functionality you can use when you read data from an Google BigQuery source for Google BigQuery and Google BigQuery V2 connectors:

Feature	Google BigQuery Connector	Google BigQuery V2 Connector
Custom query	No	Yes
Legacy SQL for custom query	No	Yes
Region ID	No	Yes
Allow Large Results	Yes	Yes
Query results destination table	Yes	Yes
Persist query results destination table	Yes	Yes
Job poll interval	Yes	Yes
Read mode - direct	Yes	Yes
Read mode - staging	Yes	Yes
Local staging file directory	Yes	Yes
Multiple threads for downloading staging files	Yes	Yes
Staging file compression	Yes	Yes
Pre-SQL and pre-SQL configurations	Yes	Yes

Feature	Google BigQuery Connector	Google BigQuery V2 Connector
Post-SQL and post-SQL configurations	Yes	Yes
SQL override query	No	Yes
Legacy SQL for SQL override query	No	Yes
Retry Options	No	Yes

Target functionality

When you import an object from Google BigQuery to write data, you can configure the advance target properties to determine the write operation behavior. For example, you can write data in streaming mode or you can retain Google Cloud Storage staging files after the write operation is complete using bulk mode.

The following table lists the target functionality you can use when you write data to a Google BigQuery target:

Target functionality	Google BigQuery mapping	Google BigQuery V2 mapping
Write mode - bulk	Yes	Yes
Specify optional custom properties in connection	No	Yes
Create disposition	Yes	Yes
Write disposition	Yes	Yes
Region ID	No	Yes
Staging file in Google Cloud Storage	Yes	Yes
Persist staging file after writing to Google BigQuery	Yes	Yes
Staging file directory	Yes	Yes
Write mode - streaming	Yes	Yes
Write mode - CDC	No	Yes
Use merge query for update, upsert, and delete operations	No	Yes
Streaming template table suffix	Yes	Yes
Rows per streaming request	Yes	Yes

Target functionality	Google BigQuery mapping	Google BigQuery V2 mapping
Staging file compression	Yes	Yes
Job poll interval	Yes	Yes
Multiple threads for uploading staging file	Yes	Yes
Allow quoted newlines	Yes	Yes
Field delimiter	Yes	Yes
Quote character	Yes	Yes
Allow jagged rows	Yes	Yes
Pre-SQL and pre-SQL configurations	Yes	Yes
Post-SQL and post-SQL configurations	Yes	Yes
Enable Merge	No	Yes
Update Override	No	Yes
Truncate target table	No	Yes

CHAPTER 3

Task examples

This section lists all the task examples that Google BigQuery and Google BigQuery V2 Connectors support.

Synchronization task use case

You work for an e-commerce organization that stores sales order details in a MySQL database. Your organization needs to move the data from the MySQL database to an Google BigQuery target.

Use Google BigQuery Connector to create a synchronization task to write to an Google BigQuery target.

Mapping and mapping task use case

You work for an organization that stores purchase order details, such as customer ID, item codes, and item quantity in an on-premise MySQL database. You need to analyze purchase order details to know the items ordered in a particular state and move data from the on-premise MySQL database to state-wise target tables in an affordable cloud-based environment.

Use Google BigQuery V2 Connector to create a mapping to state-wise read purchase records from the MySQL database and write them to multiple Google BigQuery targets to prepare an upcoming marketing campaign for all states.

Mapping task with Oracle CDC sources use case

Your organization needs to replicate real-time changed data from a mission-critical Oracle production system to minimize intrusive, non-critical work, such as offline reporting or analytical operations system.

Use Google BigQuery V2 Connector to capture changed data from the Oracle CDC source and write the changed data to an Google BigQuery target table. Add the Oracle CDC sources in mappings, and then run the associated mapping tasks to write the changed data to the target.

Mapping in advanced mode use case

You work for an organization that stores large amount of purchase order details, such as customer ID, item codes, and item quantity in Google Cloud Storage. You need to port the data from Google Cloud Storage to another cloud-based environment to quickly analyze the purchase order details and to increase future revenues.

Use Google BigQuery V2 Connector to create a mapping in advanced mode to achieve faster performance when you read all the purchase records from Google Cloud Storage and write the records to a Google BigQuery target.

Part II: Data Integration with Google BigQuery V2 Connector

This part contains the following chapters:

CHAPTER 4

Introduction to Google BigQuery V2 Connector

You can use Google BigQuery V2 Connector to securely read data from or write data to Google BigQuery. You can read from or write to Google BigQuery tables. You can also read from standard and materialized views.

When you use Google BigQuery V2 Connector, you can create a Google BigQuery V2 connection and use the connection in Data Integration mappings and tasks.

You can switch mappings to advanced mode to include transformations and functions that enable advanced functionality.

When you run a task or mapping, the Secure Agent uses the JAVA client libraries of the Google APIs to integrate with Google BigQuery.

Google BigQuery V2 Connector assets

Create assets in Data Integration to integrate data using Google BigQuery V2 Connector.

When you use Google BigQuery V2 Connector, you can include the following Data Integration assets:

- Data transfer task
- Dynamic mapping task
- Mapping
- Mapping task

For more information about configuring assets and transformations, see *Mappings*, *Transformations*, and *Tasks* in the Data Integration documentation.

Google BigQuery example

Your organization is an open source log data collector, which collects log data from multiple sources and unifies them.

Logs help you understand how systems and applications perform. As the scale and complexity of the system increases, it is difficult to manage multiple logs from different sources.

To overcome this problem, you can use Google BigQuery V2 Connector to write data to a Google BigQuery target and query terabytes of logs in seconds. You can then use the data to fix and improve the system performance in near real time.

Administration of Google BigQuery V2 Connector

Google BigQuery is a RESTful web service that the Google Cloud Platform provides.

The Google BigQuery Connector uses the following Google APIs to integrate with Google BigQuery:

- google-api-services-bigquery-v2-rev20220827-2.0.0
- google-cloud-bigquery-2.16.0

Before you use Google BigQuery V2 Connector, you must complete the following prerequisite tasks:

- Ensure you have the project ID, dataset ID, source table name, and target table name when you create mappings in Data Integration.
- Verify that you have read and write access to the Google BigQuery dataset that contains the source table and target table.
- When you read data from or write data to a Google BigQuery table, you must have the required permissions to run the mapping successfully.
- If your organization passes data through a proxy, virtual private cloud, or protective firewall, you must configure your firewall to allow the `www.googleapis.com` and `www.accounts.google.com` URI for Google BigQuery V2 Connector to transfer data through a proxy, virtual private cloud, or firewall.
- If you use bulk mode to write data to Google BigQuery, verify that you have write access to the Google Cloud Storage path where the Secure Agent creates the staging file.
- If you use DTM staging mode to write data to Google BigQuery, ensure that you configure the Secure Agent to enable the DTM staging mode.
- If you use staging mode to read data from Google BigQuery, verify that you have read access to the Google Cloud Storage path where the Secure Agent creates the staging file to store the data from the Google BigQuery source.
- To read or write Avro, Parquet, or JSON files, verify that your organization does not have more than one Cloudera 6.1 distribution enabled.

C H A P T E R 5

Google BigQuery V2 connections

Create a Google BigQuery V2 connection to securely read data from or write data to Google BigQuery.

You can use a Google BigQuery V2 connection to specify sources, targets, and lookups in mappings and mapping tasks. You can also use the Google BigQuery V2 connection in an SQL transformation.

Connect to Google BigQuery

Let's configure the Google BigQuery V2 connection properties to connect to Google BigQuery.

Before you begin

Before you configure a connection, ensure that you download the Google service account key file in JSON format. The service account key file is created when you create a Google service account.

You require the client email, private key, and project ID from the service account key JSON file to create a Google BigQuery connection.

The following video shows you how to get the information you need from your Google BigQuery account:

Connection details

The following table describes the basic connection properties:

Property	Description
Connection Name	Name of the connection. Each connection name must be unique within the organization. Connection names can contain alphanumeric characters, spaces, and the following special characters: _ . + -, Maximum length is 255 characters.
Description	Description of the connection. Maximum length is 4000 characters.

Property	Description
Type	Google BigQuery V2
Runtime Environment	The name of the runtime environment where you want to run tasks. Select a Secure agent, Hosted Agent, or serverless runtime environment.
Service Account Email	The client_email value from the Google service account key JSON file.
Service Account Key	The private_key value from the Google service account key JSON file.
Project ID	The project_id value from the Google service account key JSON file. If you have created multiple projects with the same service account, enter the ID of the project that contains the dataset that you want to connect to.

Note: If you want to validate the credentials for the Service Account Email, Service Account Key, and Project ID during a test connection, set the flag `CredentialValidation:true` in the **Provide Optional Properties** field in advanced settings.

Advanced settings

The following table describes the advanced connection properties:

Property	Description
Enable BigQuery Storage API	Select this option to use Google BigQuery Storage to stage the files when you read or write data. Default is unselected.
Storage Path	Path in Google Cloud Storage where the agent creates a local stage file to store the data temporarily. The agent uses this storage when it reads data in staging mode or writes data in bulk mode. Use one of the following formats: - gs://<bucket_name> - gs://<bucket_name>/<folder_name> This property is not applicable if you use Google BigQuery Storage to stage the files.
Connection Mode	The mode that you want to use to read data from or write data to Google BigQuery. Select one of the following connection modes: - Simple. Flattens each field within the Record data type field as a separate field in the mapping. - Hybrid[1]. Displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery V2 Connector displays the top-level Record data type field as a single field of the String data type in the mapping. - Complex[1]. Displays all the columns in the Google BigQuery table as a single field of the String data type in the mapping. Default is Simple. This property is applicable if you use Google Cloud Storage to stage the files.
Use Legacy SQL for Custom Query[1]	Select this option to use legacy SQL to define a custom query. If you clear this option, use standard SQL to define a custom query. This property is applicable if you use Google Cloud Storage to stage the files. This property doesn't apply if you configure the Google BigQuery V2 connection in hybrid or complex mode.

Property	Description
Dataset Name for Custom Query[1]	When you define a custom query, specify a Google BigQuery dataset.
Schema Definition File Path[1]	Directory on the Secure Agent machine where the Secure Agent creates a JSON file with the sample schema of the Google BigQuery table. The JSON file name is the same as the Google BigQuery table name. Alternatively, you can specify a storage path in Google Cloud Storage where the Secure Agent creates a JSON file with the sample schema of the Google BigQuery table. You can download the JSON file from the specified storage path in Google Cloud Storage to a local machine. The schema definition file is required if you configure complex connection mode in the following scenarios: - You add a Hierarchy Builder transformation in a mapping to read data from relational sources and write data to a Google BigQuery target. - You add a Hierarchy Parser transformation in a mapping to read data from a Google BigQuery source and write data to relational targets. When you use a serverless runtime environment, specify a storage path in Google Cloud Storage. This property is applicable if you use Google Cloud Storage to stage the files.
Region ID	The region name where the Google BigQuery dataset that you want to access resides. **Note:** Ensure that you specify a bucket name or the bucket name and folder name in the **Storage Path** property that resides in the specified region. For more information about the regions supported by Google BigQuery, see Dataset locations.
Staging Dataset[1]	The Google BigQuery dataset name where you want to create the staging table to stage the data. You can define a Google BigQuery dataset that is different from the source or target dataset. This property is applicable if you use Google Cloud Storage to stage the files.
Provide Optional Properties[1]	Comma-separated key-value pairs of custom properties in the Google BigQuery V2 connection to configure certain source and target functionalities. For more information about the list of custom properties that you can specify, see Optional Properties configuration Knowledge Base.
Enable Retry[1]	Select this option if you want the Secure Agent to attempt a retry to receive the response from the Google BigQuery endpoint. You can configure the retry strategy to read data from Google BigQuery in direct or staging mode and write data to Google BigQuery in bulk mode. The retry strategy is not applicable in the CDC and streaming modes when you write data to a Google BigQuery target. Default is unselected.
Maximum Retry Attempts	Appears only if you select the **Enable Retry** property. The maximum number of retry attempts that the Secure Agent performs to receive the response from the Google BigQuery endpoint. If the Secure Agent fails to connect to Google BigQuery within the maximum retry attempts, the connection fails. Default is 6 attempts.
Initial Retry Delay	Appears only if you select the **Enable Retry** property. The initial wait time in seconds before the Secure Agent attempts to retry the connection. Default is 1 second.

Property	Description
Retry Delay Multiplier	Appears only if you select the **Enable Retry** property. The multiplier that the Secure Agent uses to exponentially increase the wait time between successive retry attempts up to the maximum retry delay time. Default multiplier is 2.0. You can also use fractional values.
Maximum Retry Delay	Appears only if you select the **Enable Retry** property. The maximum wait time in seconds that the Secure Agent waits between successive retry attempts. Default is 32 seconds.
Total Timeout	Appears only if you select the **Enable Retry** property. The total time duration in seconds that the Secure Agent attempts to retry the connection after which the connection fails. Default is 50 seconds.
[1] Doesn't apply to mappings in advanced mode.	

Proxy server settings

If your organization uses an outgoing proxy server to connect to the Internet, the Secure Agent connects to Informatica Intelligent Cloud Services through the proxy server.

You can configure the Secure Agent and the serverless runtime environment to use the proxy server on Windows and Linux. You can use the unauthenticated or authenticated proxy server. The proxy settings applies to connections used in mappings and in mappings in advanced mode.

Use one of the following methods to configure the proxy settings:

- Configure the Secure Agent through the Secure Agent Manager on Windows or shell command on Linux.

  For instructions, see "Configure the proxy settings on Windows" or "Configure the proxy settings on Linux" in *Getting Started* in the Data Integration help .

- Configure the JVM options for the DTM in the Secure Agent properties. For instructions, see the Proxy server settings Knowledge Base article.

To configure the proxy settings for the serverless runtime environment, see "Using a proxy server" in *Runtime Environments* in the Administrator help.

Configure proxy settings for NTLM authentication

You can use a proxy server that uses NTLM authentication to connect to Google BigQuery. To configure the proxy settings for NTLM authentication, perform the following steps:

1. In Administrator, select **Runtime Environments**.
2. Select the Secure Agent for which you want to configure from the list of available Secure Agents.
3. In the upper-right corner, click **Edit**.
4. In the **System Configuration Details** section, select the **Type** as **DTM** for the Data Integration Server.
5. Edit the **JVMOption1** and add the following value:

```
-Dhttp.auth.ntlm.domain=<domain name>
```

6. Select the **Type** as **Platform** for the Data Integration Server.

7. Edit the **INFA_DEBUG** property and add the following value:
```
-Dhttp.auth.ntlm.domain=<domain name>
```

8. Click **Save**.

9. Restart the Secure Agent.

CHAPTER 6

Mappings for Google BigQuery V2

When you configure a mapping, you describe the flow of data from the source to the target.

A mapping defines reusable data flow logic that you can use in mapping tasks.

When you create a mapping, you define the Source, Target, and Lookup transformations to represent a Google BigQuery V2 object. Use the Mapping Designer in Data Integration to add the Source, Target, or Lookup transformations in the mapping canvas and configure the Google BigQuery V2 source, target, and lookup properties.

In advanced mode, the Mapping Designer updates the mapping canvas to include transformations and functions that enable advanced functionality.

The Google BigQuery V2 Connector uses Spark BigQuery connector library to write the data in a mapping in advanced mode. This library utilizes the Google BigQuery Storage API to directly write the data into Google BigQuery and does not use Google Cloud Storage for intermediate staging.

You can use Monitor to monitor the jobs.

Google BigQuery Storage API

When you configure a connection, you can choose to use Google BigQuery Storage to staging when you read or write data. To use Google BigQuery Storage, select the **Enable BigQuery Storage API** connection advanced property.

You can use hybrid connection mode to import the metadata and stage files that contain hierarchical data types such as record and repeat.

You can use the standard SQL format for the custom query, but you cannot use the legacy SQL format.

Read operation

- Uses the Staging mode as read mode.
- You can configure the following runtime attributes for the read operation:
    - Connection
    - Source Type
    - Object
    - Parameter
    - Query
    - Filter

- Source Dataset ID
- Source Table Name
- Source Staging Dataset
- Number of Rows to Read
- Allow Large Results
- Query Results Table Name
- Job Poll Interval in Seconds
- Read Mode
- Persist Destination Table
- pre SQL
- pre SQL Configuration
- post SQL
- post SQL Configuration
- SQL Override Query
- Use Legacy SQL for SQL Override

Write operation

- Uses the Bulk mode as write mode.

- You cannot use update, delete, and merge target operations on the rows in a target table that were written in the last 30 minutes. If you configure a mapping with this scenario, the mapping fails with one of the following errors:

  - [ERROR] The Google BigQuery V2 Target definition post-SQL operation failed with the following error: [UPDATE or DELETE statement over table table1_BQ_Storage.tgt_mct_151_table1 would affect rows in the streaming buffer, which is not supported].

  - [ERROR] Error occured while trying to Initialize Data Source Operation | com.informatica.cci.runtime.internal.utils.impl.CExceptionImpl: Unable to render embedded object: File (Truncate Target Failed UPDATE or DELETE statement over table automation-bigquery-project.table1_BQ_Storage.tgt_mct_151_table1 would affect rows in the streaming buffer, which is not supported) not found.

- You cannot use the `write empty` option in the **Write Disposition** property.

- You can use the `write truncate` option in the **Write Disposition** property only with a truncated target table.

- You can configure the following runtime attributes for the write operation:
  - Connection
  - Object
  - Target Type. You can use single object as the Target Type.
  - Operation. You can use only the Insert operation.
  - Write Mode
  - Create New at Runtime
  - Data Driven Condition
  - Update Columns

- Target Dataset ID

- Update Mode

- Enable Data Driven

- Enable Merge

- Update Override

- Target Table Name

- Target Staging Dataset

- Create Disposition

- Job Poll Interval in Seconds

- Pre SQL

- Pre SQL Configuration

- Post SQL

- Suppress post SQL on Error

- Post SQL Configuration

- Truncate target table

Mappings with different connection modes

You can configure a Google BigQuery V2 connection to use one of the following connection modes:

Simple mode

If you use simple mode, Google BigQuery V2 Connector flattens each field within the Record data type field as a separate field in the field mapping.

Hybrid mode

If you use hybrid mode, Google BigQuery V2 Connector displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery V2 Connector displays the top-level Record data type field as a single field of the String data type in the field mapping.

Complex mode

If you use complex mode, Google BigQuery displays all the columns in the Google BigQuery table as a single field of the String data type in the field mapping.

Google BigQuery V2 Connector reads and writes the Google BigQuery data based on the connection mode that you configure for the Google BigQuery V2 connection.

You have a Customers table in Google BigQuery that contains primitive fields and the **Address** field of the Record data type. The Address field contains two primitive sub-fields, **City** and **State**, of the String data type.

The following image shows the schema of the Customers table in Google BigQuery:

ID	INTEGER	NULLABLE
Name	STRING	NULLABLE
Address	RECORD	NULLABLE
Address.City	STRING	NULLABLE
Address.State	STRING	NULLABLE
Mobile	STRING	REPEATED
Totalpayments	FLOAT	NULLABLE
age	INTEGER	REPEATED

The following table shows the Customers table data in Google BigQuery:

ID	Name	Address.City	Address.State	Mobile	Totalpayments
14	John	LOS ANGELES	CALIFORNIA	+1-9744884744	18433.90
				+1-8267389993	
29	Jane	BOSTON	MANHATTAN	+1-8789390309	28397.33
				+1-9876553784	
				+1-8456437848	

Simple mode

If you use simple connection mode, Google BigQuery V2 Connector flattens each field within the Record data type field as a separate field in the **Field Mapping** tab.

The following table shows two separate fields, Address_City and Address_State, for the respective sub-fields within the Address Record field in the Customers table:

ID	Name	Address_City	Address_State	Mobile	Totalpayments
14	John	LOS ANGELES	CALIFORNIA	+1-9744884744	18433.90
14	John	LOS ANGELES	CALIFORNIA	+1-8267389993	18433.90
29	Jane	BOSTON	MANHATTAN	+1-8789390309	28397.33
29	Jane	BOSTON	MANHATTAN	+1-9876553784	28397.33
29	Jane	BOSTON	MANHATTAN	+1-8456437848	28397.33

The following image shows the fields in the **Field Mapping** tab of the Target transformation:

Hybrid mode

If you use hybrid connection mode, Google BigQuery V2 Connector displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery V2 Connector displays the top-level Record data type field as a single field of the String data type in the **Field Mapping** tab.

The following image shows the **Field Mapping** tab of the Target transformation:

Complex mode

If you use complex connection mode, Google BigQuery V2 Connector displays all the columns in the Google BigQuery table as a single field of the String data type in the **Field Mapping** tab.

The following image shows the STRING_DATA field in the **Field Mapping** tab of the Target transformation:

Rules and guidelines for Google BigQuery V2 connection modes

Simple mode

Consider the following rules and guidelines when you configure a Google BigQuery V2 connection to use simple connection mode:

- You cannot configure mappings in advanced mode.
- You can read data from a repeated column from a Google BigQuery source table only when you select **Direct** as the **Read Mode**.
- You cannot create a Google BigQuery target table that contains repeated columns using the **Create Target** option.
- If the Google BigQuery source table contains repeated columns, you cannot configure data filters for these columns.
- If the Google BigQuery table contains more than one repeated column, you cannot preview data.
- If the Google BigQuery target table contains a repeated column of the Record data type, you cannot configure update, upsert, and delete operations for these columns.
- You can use CSV format as the data format of the staging file only when the Google BigQuery table does not contain columns of the Record data type or repeated columns.
- If the Google BigQuery target table contains columns of the Record data type and repeated columns, you cannot configure update, upsert, and delete operations for these columns when you do not use the Merge query.

- When you read data from a Google BigQuery source, you must not map more than one repeated column in a single mapping. You must create multiple mappings for each repeated column.
- You cannot import multiple source tables in a Source transformation.

Hybrid mode

Consider the following rules and guidelines when you configure a Google BigQuery V2 connection to use hybrid connection mode:

- You cannot preview data.
- You cannot use a legacy SQL statement to define a custom query. You must use a standard SQL to define a custom query
- If the Google BigQuery source table contains columns of the Record data type and repeated columns, you cannot configure data filters for these columns.
- When you do not use the Merge query and the key field is a column of the Record data type or a repeated column, you cannot configure update, upsert, and delete operations.
- You must select JSON (Newline Delimited) format as the data format of the staging file under the advanced target properties. You can use CSV format as the data format of the staging file only when the Google BigQuery table does not contain columns of the Record data type or repeated columns.
- The following CSV formatting options in the advanced target properties are not applicable:
  - Allow Quoted Newlines
  - Field Delimiter
  - Allow Jagged Rows

Complex mode

Consider the following rules and guidelines when you configure a Google BigQuery V2 connection to use complex connection mode:

- You cannot configure mappings in advanced mode.
- You cannot import multiple source tables in a Source transformation.
- You cannot preview data.
- You cannot use a legacy SQL statement to define a custom query. You must use a standard SQL to define a custom query
- You cannot create a Google BigQuery target table using the **Create Target** option.
- You cannot truncate the Google BigQuery target table before loading data to the target using the **Truncate target table** option.
- When you configure a Google BigQuery source connection to use complex connection mode, you cannot configure data filters for the source.
- You cannot configure update, upsert, and delete operations.
- You must select JSON (Newline Delimited) format as the data format of the staging file under the advanced target properties.
- You cannot use CSV format as the data format of the staging file. The following CSV formatting options in the advanced target properties are not applicable:
  - Allow Quoted Newlines
  - Field Delimiter
  - Allow Jagged Rows
- You cannot use key range partitioning for Google BigQuery sources.

Google BigQuery V2 sources in mappings

To read data from Google BigQuery, configure a Google BigQuery object as the Source transformation in a mapping.

Specify the name and description of Google BigQuery source. Configure the source and advanced properties for the source object in mappings.

The following table describes the source properties that you can configure for a Google BigQuery source:

Property	Description
Connection	Name of the Google BigQuery V2 source connection. Select a source connection, or click **New Parameter** to define a new parameter for the source connection. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option when you create a parameter.
Source Type	Type of the Google BigQuery source object. Select **Single Object**, **Multiple Objects¹**, **Query¹** or **Parameter**. When you select single object as the source type, you can choose a table or view. For the other source object types, you can choose a table.
Object	Name of the Google BigQuery source object based on the source type selected.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the source object or click **New Parameter** to define a new parameter for the source object. The Parameter property appears only if you select Parameter as the source type. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option when you create a parameter. When the task runs, the agent uses the parameters from the file that you specify in the task advanced session properties.
Query¹	Click on **Define Query** and enter a valid custom query. The **Query** property appears only if you select **Query** as the source type. You can parameterize a custom query object at runtime in a mapping. Select the source advanced property **Use EXPORT DATA Statement to stage** to use the ORDER BY clause in a custom query in staging mode.
Filter	Configure a simple filter or an advanced filter to remove rows at the source. You can improve efficiency by filtering early in the data flow. A simple filter includes a field name, operator, and value. Use an advanced filter to define a more complex filter condition, which can include multiple conditions using the AND or OR logical operators. Only simple filter is applicable for mappings in advanced mode.

¹ Doesn't apply to mappings in advanced mode.

The following table describes the advanced properties that you can configure for a Google BigQuery source:

Property	Description
Source Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the Source transformation.
Source Table Name	Optional. Overrides the Google BigQuery table name that you specified in the Source transformation.
Source Staging Dataset[1]	Optional. Overrides the Google BigQuery staging dataset name that you specified in the connection and the Source Dataset ID source advanced property.
Number of Rows to Read	Specifies the number of rows to read from the Google BigQuery source table.
Allow Large Results[1]	Determines whether Google BigQuery V2 Connector must produce arbitrarily large result tables to query large source tables. If you select this option, you must specify a destination table to store the query results.
Query Results Table Name[1]	Required if you select the **Allow Large Results** option. Specifies the destination table name to store the query results. If the table is not present in the dataset, Google BigQuery V2 Connector creates the destination table with the name that you specify.
Job Poll Interval in Seconds[1]	The number of seconds after which Google BigQuery V2 Connector polls the status of the read job operation. Default is 10.
Read Mode	Specifies the read mode to read data from the Google BigQuery source. You can select one the following read modes: - Direct. In direct mode, Google BigQuery V2 Connector reads data directly from the Google BigQuery source table. **Note:** When you use hybrid and complex connection mode, you cannot use direct mode to read data from the Google BigQuery source. - Staging[1]. In staging mode, Google BigQuery V2 Connector exports data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery V2 Connector downloads the data from Google Cloud Storage into the local stage file and then reads data from the local stage file. Default is Direct mode.
Use EXPORT DATA Statement to stage	Indicates whether to support ORDER BY clause in a custom query or SQL override query. This property applies to staging mode. You can choose a table or view to stage the data.
Number of Threads for Downloading Staging Files[1]	Specifies the number of files that Google BigQuery V2 Connector downloads at a time to enable parallel download. This property applies to staging mode.
Local Stage File Directory[1]	Specifies the directory on your local machine where Google BigQuery V2 Connector stores the Google BigQuery source data temporarily before it reads the data. This property applies to staging mode. **Note:** This property is not applicable when you use a serverless runtime environment.

Property	Description
Staging File Name[1]	Name of the staging file where data from the Google BigQuery source table is exported to Google Cloud Storage. This property applies to staging mode.
Data Format of the staging file[1]	Specifies the data format of the staging file. You can select one of the following data formats: - Avro - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ. - Parquet This property applies to staging mode.
Enable Staging File Compression[1]	Indicates whether to compress the size of the staging file in Google Cloud Storage before Google BigQuery V2 Connector reads data from the staging file. You can enable staging file compression to reduce cost and transfer time. This property applies to staging mode.
Persist Extract Staging File After Download[1]	Indicates whether Google BigQuery V2 Connector must persist the staging file after it reads data from the staging file. By default, Google BigQuery V2 Connector deletes the staging file.
Persist Destination Table[1]	Indicates whether Google BigQuery V2 Connector must persist the query results table after it reads data from the query results table. By default, Google BigQuery V2 Connector deletes the query results table.
pre SQL[1]	SQL statement that you want to run before reading data from the source. For example, if you want to select records in the database before you read the records from the table, specify the following pre SQL statement: `SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;`

pre SQL Configuration[1]	Specify a pre SQL configuration.
	For example,
	`DestinationTable:PRESQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

post SQL[1]	SQL statement that you want to run after reading data from the source.
	For example, if you want to update records in a table after you read the records from a source table, specify the following post SQL statement:
	`UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	`SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where`
	`fullname='John Doe'`

post SQL Configuration[1]	Specify a post SQL configuration.
	For example,
	`DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:True,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

Property	Description
SQL Override Query[1]	Overrides the default SQL query used to read data from the Google BigQuery source. **Note:** When you specify SQL override query, you must specify a dataset name in the Source Dataset ID advanced source property. Ensure that the list of selected columns, data types, and the order of the columns that appear in the query matches the columns, data types, and order in which they appear in the source object. Ensure that you only map all the columns in the SQL override query to the target. Does not apply when you enable partitioning. Select the source advanced property **Use EXPORT DATA Statement to stage** to use the ORDER BY clause in a SQL Override Query in staging mode. When staging optimization is enabled in a mapping, the columns mapped in the SQL Override Query must match the columns in the source object.

Use Legacy SQL for SQL Override[1]	Indicates that the SQL Override query is specified in legacy SQL.
	Use the following format to specify a legacy SQL query for the **SQL Override Query** property:
	`SELECT <Col1, Col2, Col3> FROM [projectID:datasetID.tableName]`
	Clear this option to define a standard SQL override query.
	Use the following format to specify a standard SQL query for the **SQL Override Query** property:
	`SELECT * FROM \`projectID.datasetID.tableName\``

Retry Options[1]	Comma-separated list to specify the following retry options: - Retry Count. The number of retry attempts to read data from Google BigQuery. - Retry Interval. The time in seconds to wait between each retry attempt. - Retry Exceptions. The list of exceptions separated by pipe (|) character for which the retries are made. Use the following format to specify the retry options: For example, `RetryCount:5,RetryInterval:1,RetryExceptions:java.net.ConnectException|java.io.IOException` **Note:** The retry options are available for preview. Preview functionality is supported for evaluation purposes but is unwarranted and is not production-ready. Informatica recommends that you use in non-production environments only. Informatica intends to include the preview functionality in an upcoming release for production use, but might choose not to in accordance with changing market or technical circumstances. For more information, contact Informatica Global Customer Support. To use the functionality, your organization must have the appropriate licenses.
Number of Spark Partitions[2]	Specifies the maximum number of partitions that the Spark engine splits the data into. Default is 1.
Label[1]	A label is a key-value pair that you can assign to a Google BigQuery resource. You can attach a label for each transformation. Use labels to organize and filter the associated jobs in the Google Cloud Platform Log Explorer. For more information about labels and their usage requirements, see "Assign label to the Google BigQuery resources" on page 73.
Billing Project ID[1]	The project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs query and extract jobs. If you omit the project ID here, the Secure Agent runs query and extract jobs in the Google Cloud project corresponding to the **Project ID** value specified in the Google BigQuery V2 connection.
[1] Doesn't apply to mappings in advanced mode. [2] Applies only to mappings in advanced mode.	

You can set the tracing level in the advanced properties session to determine the amount of details that logs contain.

The following table describes the tracing levels that you can configure:

Property	Description
Terse	The Secure Agent logs initialization information, error messages, and notification of rejected data.
Normal	The Secure Agent logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes session results, but not at the level of individual rows.
Verbose Initialization	In addition to normal tracing, the Secure Agent logs additional initialization details, names of index and data files used, and detailed transformation statistics.
Verbose Data	In addition to verbose initialization tracing, the Secure Agent logs each row that passes into the mapping. Also notes where the Secure Agent truncates string data to fit the precision of a column and provides detailed transformation statistics. When you configure the tracing level to verbose data, the Secure Agent writes row data for all rows in a block when it processes a transformation.

Adding multiple source objects

When you create a Source transformation, you can select Google BigQuery V2 multiple object as the source type and then configure a join to combine the tables. You can define an advanced relationship or a query to join the tables. You must use the standard SQL to define the query to join the tables.

1. In the Source transformation, click the **Source Type** as **Multiple Objects**.
2. From the **Actions** menu, click **Add Source Object**.
3. Select the source object that you want to add from the displayed list and click **OK**.
4. From the **Related Objects Actions** menu, select **Advanced Relationship**.
5. In the **Advanced Relationship** window, you can click **Add Object** to add more objects.
6. Set your own conditions or specify a query to define the relationship between the tables.

   Note: When you configure a join expression, select the fields and define a join condition or a query syntax. You must prefix the Project ID before the Dataset ID in the join condition to combine multiple tables. Use the following example to configure the join condition or join query with the Project ID prefix:

   - Join condition:

     ```
     `P1.D3.T3`.col5 = `P1.D2.T2`.col3 ON `P1D2.T2`.col4 = `P1.D1.T1`.col2
     ```

   - Join query:

     ```
     `P1.D1.T1` LEFT OUTER JOIN `P1.D2.T2` FULL OUTER JOIN `P1.D3.T3` ON `P1.D3.T3`.col5 = `P1.D2.T2`.col3 ON `P1.D2.T2`.col4 = `P1.D1.T1`.col2
     ```

   In the example, `P` represents the Project ID, `D` represents the Dataset ID, and `T` represents the Table Name.

   Note: If you configure a filter, prefix the Project ID before the Dataset ID in the filter condition. For example, provide the simple filter condition as, `` `Project.Dataset.Table`.column ``

   Note: If you specify a SQL override query for multiple source tables, you must use the following format for the SQL override query: `select `project_id.dataset.table`.col1 AL_dataset_table_col1,`project_id.dataset.table`.col2 AL_dataset_table_col2,`

```
`project_id.dataset1.table1`.col1
AL_dataset1_table1_col1,`project_id.dataset1.table1`.col2 AL_dataset1_table1_col2 from
`project_id.dataset.table` <join condition> `project_id.dataset1.table1` ON
`project_id.dataset.table`.col = `project_id.dataset1.table1`.col1 where <condition>
```

In the example, `AL` represents the alias prefix for the column names.

7. Click **OK**.

The following image shows an example of an advanced join condition defined between the Google BigQuery V2 tables:
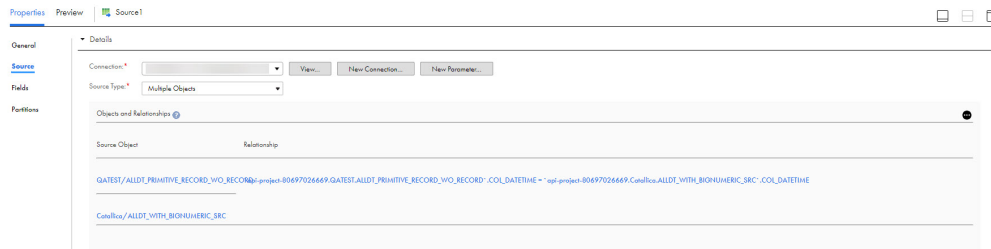
Rules and guidelines for adding multiple source objects

Consider the following rules and guidelines when you add multiple source objects:

- When you import multiple source tables, ensure that you use a Google BigQuery V2 connection in hybrid mode.
- You cannot configure partitioning when you use the advanced relationship option.
- You must not import multiple source tables with the same name from different datasets.
- You cannot use a self join when you add multiple source tables.
- When you specify a cross join condition to read data from multiple Google BigQuery tables, use the following format:
  `project_id.datasetname.tablename CROSS JOIN project_id.datasetname.tablename`
- When you configure a Filter transformation, you must not apply a filter condition with the column of Record data type in the source object.
- When you configure a Joiner or Router transformation, you must not apply a condition with the column of Record data type in the source object.
- When you configure a mapping to read data from multiple sources, ensure that the source objects does not contain columns of Record data type or Repeated columns with the same name. Otherwise, the mapping fails with the following error:
  `com.google.cloud.hadoop.repackaged.bigquery.com.google.api.client.googleapis.json.`
  `GoogleJsonResponseException: 400 Bad Request`
- When you use special characters in column names for an advanced relationship, the query generated is not valid and the mapping task fails.
- When you click **Add Object** to add more objects in the **Advanced Relationship** window, the table might fail to load or take a long time to load. If this issue occurs, import the object again.

Partitioning

When you read data from a Google BigQuery source and use simple or hybrid connection mode, you can configure key range partitioning to optimize the mapping performance at run time.

Key range partitioning

You can configure key range partitioning in a mapping that uses the simple or hybrid connection mode to read data from Google BigQuery sources. With key range partitioning, the Secure Agent distributes rows of source data based on the fields that you define as partition keys. The Secure Agent compares the field value to the range values for each partition and sends rows to the appropriate partitions.

Use key range partitioning for columns that have an even distribution of data values. Otherwise, the partitions might have unequal size. For example, a column might have 10 rows between key values 1 and 1000 and the column might have 999 rows between key values 1001 and 2000. If the mapping includes multiple sources, use the same number of key ranges for each source.

When you define key range partitioning for a column, the Secure Agent reads the rows that are within the specified partition range. For example, if you configure two partitions for a column with the ranges as 10 through 20 and 30 through 40, the Secure Agent does not read the rows 20 through 30 because these rows are not within the specified partition range.

You can also use an in-out parameter file to specify the key range values for the partition key columns in a mapping.

You can configure a partition key for fields of the following data types:

- String
- Integer
- Numeric (only if you use a Google BigQuery connection in hybrid mode)
- Timestamp
  Use the following format: YYYY-MM-DD HH24:MI:SS

Guidelines for key range partitioning in mappings

When you configure key range partitioning in a mapping, consider the following guidelines:

- You cannot configure a partition key for Record data type columns and repeated columns.
- You cannot configure key range partitioning if a row in the partition key column contains a null value.
- You cannot parameterize the value for the partition key column name in the **Partition Key** property.
- You cannot parameterize the key range values for a partition key column of the Timestamp data type.
- When you run a mapping with SQL ELT optimization, you cannot parameterize the key range values.
- You cannot use key range partitions when a mapping includes any of the following transformations:
  - Web Services
  - JSON to Relational

Configuring Key Range Partitioning

Perform the following steps to configure key range partitioning for Google BigQuery sources:

1. In the Source Properties, click the **Partitions** tab.
2. Select the required partition key from the list.

3. Click **Add New Key Range** to define the number of partitions and the key ranges based on which the Secure Agent must partition data.

   Use a blank value for the start range to indicate the minimum value. Use a blank value for the end range to indicate the maximum value.

   The following image displays the **Partitions** tab:

Google BigQuery V2 targets in mappings

To write data to a Google BigQuery target, configure a Google BigQuery object as the Target transformation in a mapping.

Specify the name and description of Google BigQuery target. Configure the target and advanced properties for the target object in mappings.

The following table describes the target properties that you can configure for a Google BigQuery target:

Property	Description
Connection	Name of the Google BigQuery V2 target connection. Select a target connection or click **New Parameter** to define a new parameter for the target connection. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option when you create a parameter. When the task runs, the agent uses the parameters from the file that you specify in the task advanced session properties.
Target Type	Type of the Google BigQuery target objects available. You can write data to a single or multiple Google BigQuery target objects. You can also parameterize the object.
Parameter	Select an existing parameter for the target object or click **New Parameter** to define a new parameter for the target object. The **Parameter** property appears only if you select Parameter as the target type. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option when you create a parameter. When the task runs, the agent uses the parameters from the file that you specify in the task advanced session properties. Does not apply when you perform a data driven operation.
Object	Name of the Google BigQuery target object based on the target type selected.

Property	Description
Create New at Runtime	Creates a target table at runtime in Google BigQuery. The target table can also contain clustered columns. Enter a name for the target object and path for the target object and select the source fields that you want to use. By default, all source fields are used. You must specify a valid dataset ID for the **Path** attribute. The target name can contain alphanumeric characters. You cannot use special characters in the file name except the underscore character (_). You cannot parameterize the target at runtime. For more information about how to create a target table with clustered columns, see "Clustering order" on page 50.
Operation	You can select one the following operations: - Insert - Update - Upsert (Update or Insert) - Delete - Data Driven **Note:** If you use complex connection mode, you cannot configure update, upsert, and delete operations.
Data Driven Condition	Flags rows for an insert, update, delete, or reject operation based on the data driven expression you specify. You must specify the data driven condition for non-CDC sources. For CDC sources, you must leave the field empty as the rows in the CDC source tables are already marked with the operation types. **Note:** Appears only when you select **Data Driven** as the operation type.
Update Columns	Specifies the temporary primary key columns to update, upsert or delete target data. If the Google BigQuery target does not include a primary key column, and the mapping performs an update, upsert, or delete task operation, click **Add** to add a temporary key. You can select multiple columns. By default, no columns are specified.

The following table describes the advanced properties that you can configure for a Google BigQuery target:

Property	Description
Target Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
UpdateMode	Determines the mode that the Secure Agent uses to update rows in the Google BigQuery target. You can select one of the following modes: - Update As Update. The Secure Agent updates all rows flagged for update if the entries exist. - Update Else Insert. The Secure Agent first updates all rows flagged for update if the entries exist in the target. If the entries do not exist, the Secure Agent inserts the entries. Default is Update as Update. Not applicable when you perform a data driven operation.
Enable Data Driven[1]	Implements data driven operation to honor flagged rows for an insert, update, delete, or reject operation based on the data driven condition. Select this option when you select **Data Driven** as the target operation.

Property	Description
Enable Merge	Implements the Merge query to perform an update, upsert, delete or data driven operation on a Google BigQuery target table. If you select the **Enable Data Driven** property, you must select this option. Default is disabled.
Use Default Column Values	Applicable when the selected data format for the staging file is CSV when the mapping contains unconnected ports. Includes the default column values for the unconnected port from the staging file to create the target. This is applicable when you have defined the default constraint value in the Google BigQuery source column. When you do not enable this option, the agent creates a target only with the connected ports. The agent populates null or empty strings for unconnected ports.

Update Override[1]	Optional. Overrides the update SQL statement that the Secure Agent generates to update the Google BigQuery target.
	To use the update override query, you must set the Operation property to **Update** and the UpdateMode property to **Update As Update**.
	Use the following format to define an update override query:
	`UPDATE `<project_name>.<dataset_name>.<table_name>` as <alias_name>`
	`SET <alias_name>.<col_name1>=:<temp_table>.<col_name1>, <alias_name>.<col_name2>=:<temp_table>.<col_name2> FROM <dataset_name>.:<temp_table> WHERE <conditional expression>`
	For example,
	`UPDATE `project1.custdataset.cust_table1` as ab`
	`SET ab.fld_str=:custtemp.fld_str, ab.fld_int=:custtemp.fld_int FROM custdataset.:custtemp WHERE ab.fld_string_req = :custtemp.fld_string_req`
	Not applicable when you perform a data driven operation.

Target Table Name	Optional. Overrides the Google BigQuery target table name that you specified in the Target transformation. **Note:** If you specify an update override query, Google BigQuery V2 Connector ignores this property.
Target Staging Dataset[1]	Optional. Overrides the Google BigQuery staging dataset name that you specified in the connection and the Target Dataset ID target advanced property.
Create Disposition	Specifies whether Google BigQuery V2 Connector must create the target table if it does not exist. You can select one of the following values: - Create if needed. If the table does not exist, Google BigQuery V2 Connector creates the table. - Create never. If the table does not exist, Google BigQuery V2 Connector does not create the table and displays an error message. Create disposition is applicable only when you perform an insert operation on a Google BigQuery target.

Property	Description
Write Disposition	Specifies how Google BigQuery V2 Connector must write data in bulk mode if the target table already exists. You can select one of the following values: - Write append. If the target table exists, Google BigQuery V2 Connector appends the data to the existing data in the table. - Write truncate. If the target table exists, Google BigQuery V2 Connector overwrites the existing data in the table. - Write empty. If the target table exists and contains data, Google BigQuery V2 Connector displays an error and does not write the data to the target. Google BigQuery V2 Connector writes the data to the target only if the target table does not contain any data. Write disposition is applicable for bulk mode. Write disposition is applicable only when you perform an insert operation on a Google BigQuery target.
Write Mode	Specifies the mode to write data to the Google BigQuery target. You can select one of the following modes: - Bulk. Google BigQuery V2 Connector first writes the data to a staging file in Google Cloud Storage. When the staging file contains all the data, Google BigQuery V2 Connector loads the data from the staging file to the BigQuery target. Google BigQuery V2 Connector then deletes the staging file unless you configure the task to persist the staging file. - Streaming[1]. Google BigQuery V2 Connector directly writes data to the BigQuery target. Google BigQuery V2 Connector writes the data into the target row by row. - CDC[1]. Applies only when you capture changed data from a CDC source. In CDC mode, Google BigQuery V2 Connector captures changed data from any CDC source and writes the changed data to a Google BigQuery target table. Default is Bulk mode. Streaming mode is not applicable when you perform a data driven operation.
Streaming Template Table Suffix[1]	Specify the suffix to add to the individual target tables that Google BigQuery V2 Connector creates based on the template target table. This property applies to streaming mode. If you select the Enable Merge option, Google BigQuery V2 Connector ignores this property. Streaming mode is not applicable when you perform a data driven operation.
Rows per Streaming Request[1]	Specifies the number of rows that Google BigQuery V2 Connector streams to the BigQuery target for each request. Default is 500 rows. The maximum row size that Google BigQuery V2 Connector can stream to the Google BigQuery target for each request is 10 MB. This property applies to streaming mode. Streaming mode is not applicable when you perform a data driven operation.
Staging file name	Name of the staging file that Google BigQuery V2 Connector creates in the Google Cloud Storage before it loads the data to the Google BigQuery target. This property applies to bulk mode.

Property	Description
Data Format of the staging file	Specifies the data format of the staging file. You can select one of the following data formats: - Avro[1] - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - Parquet[1] - CSV[1]. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ. Only JSON format is applicable for mappings in advanced mode. This property applies to bulk and CDC mode. Avro and parquet format is not applicable when you perform a data driven operation.
Persist Staging File After Loading	Indicates whether Google BigQuery V2 Connector must persist the staging file in the Google Cloud Storage after it writes the data to the Google BigQuery target. You can persist the staging file if you want to archive the data for future reference. By default, Google BigQuery V2 Connector deletes the staging file in Google Cloud Storage. This property applies to bulk mode.
Enable Staging File Compression[1]	Select this option to compress the size of the staging file before Google BigQuery writes the data to the Google Cloud Storage and decompress the staging file before it loads the data to the Google BigQuery target. You can enable staging file compression to reduce cost and transfer time.
Job Poll Interval in Seconds[1]	The number of seconds after which Google BigQuery V2 Connector polls the status of the write job operation. Default is 10.
Number of Threads for Uploading Staging file[1]	The number of files that Google BigQuery V2 Connector must create to upload the staging file in bulk mode.
Local Stage File Directory[1]	Specifies the directory on your local machine where Google BigQuery V2 Connector stores the files temporarily before writing the data to the staging file in Google Cloud Storage. This property applies to bulk mode. **Note:** This property is not applicable when you use a serverless runtime environment.
Allow Quoted Newlines[1]	Indicates whether Google BigQuery V2 Connector must allow the quoted data sections with newline character in a .csv file.
Field Delimiter[1]	Indicates whether Google BigQuery V2 Connector must allow field separators for the fields in a .csv file.
Quote Character[1]	Specifies the quote character to skip when you write data to Google BigQuery. When you write data to Google BigQuery and the source table contains the specified quote character, the task fails. Change the quote character value to a value that does not exist in the source table. Default is double quotes.
Allow Jagged Rows[1]	Indicates whether Google BigQuery V2 Connector must accept the rows without trailing columns in a .csv file.

Property	Description
Pre SQL[1]	SQL statement that you want to run before writing data to the target. For example, if you want to select records from the database before you write the records into the table, specify the following pre-SQL statement: `SELECT * FROM 'api-project-80697026669.EMPLOYEE.RegionNation' LIMIT 1000`

Pre SQL Configuration[1]	Specify a pre-SQL configuration.
	
	 For example,
	
	 `DestinationTable:PRESQL_TGT2,`
	 `DestinationDataset:EMPLOYEE,`
	 `FlattenResults:False,`
	 `WriteDisposition:WRITE_TRUNCATE,`
	 `UseLegacySql:False`

Post SQL[1]	SQL statement that you want to run after writing the data into the target.
	
	 For example, if you want to update records in a table after you write the records into the target table, specify the following post-SQL statement:
	
	 `UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	 `SET phoneNumber.number =1000011, phoneNumber.areaCode=100 where fullname='John Doe'`

Suppress post SQL on Error[1]	Indicates whether the Secure Agent must abort the post-SQL query execution in case the task fails to write data to the Google BigQuery target table due to errors. Default is disabled.

Post SQL Configuration[1]	Specify a post-SQL configuration.
	
	 For example,
	
	 `DestinationTable:POSTSQL_SRC,`
	 `DestinationDataset:EMPLOYEE,`
	 `FlattenResults:True,`
	 `UseLegacySQL:False`

Truncate target table	Truncates the Google BigQuery target table before loading data to the target. Default is disabled.
Allow Duplicate Inserts	Applicable when the selected data format for the staging file is CSV when the mapping contains both connected and unconnected ports. Includes the values from the connected ports and considers the default column values from the unconnected ports from the staging file while loading to the target. If the ports are unconnected, the agent considers the default value only if you have defined the default constraint value in the Google BigQuery target table. When you do not enable the **Use Default Column Values** option, the agent populates the target with values from the connected ports. This property doesn't apply when you create a new target at runtime. Also, when the selected write disposition selected is **Write Truncate**, the agent removes the field from the target table that is not part of the schema and removes the default constraint for the target column.
Disable Duplicate Update Rows[1]	Determines if multiple incoming rows attempt to update the same target row, the Secure Agent must process only one of the incoming rows and ignore the rest of the incoming rows. Select this option to configure the mapping to process only one of the incoming rows and ignore the rest of the incoming rows. Default is disabled.

Property	Description
Forward Rejected Rows	Applicable only when you configure DD_REJECT constant in the data driven condition to reject all the rows. Otherwise, this property is not applicable for Google BigQuery V2 Connector.
Label[1]	A label is a key-value pair that you can assign to a Google BigQuery resource. You can attach a label for each transformation. Use labels to organize and filter the associated jobs in the Google Cloud Platform Log Explorer. For more information about labels and their usage requirements, see "Assign label to the Google BigQuery resources" on page 73.
Billing Project ID	The project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs query and load jobs. If you omit the project ID here, the Secure Agent runs query and load jobs in the Google Cloud project corresponding to the **Project ID** value specified in the Google BigQuery V2 connection.

[1]Doesn't apply to mappings in advanced mode.

Mapping tasks with CDC sources

You can use Google BigQuery V2 Connector to capture changed data from any CDC source and write the changed data to a Google BigQuery target. Add the CDC sources in mappings, and then run the associated mapping tasks to write the changed data to the target. When you capture changed data from a CDC source, you can only configure a single Google BigQuery V2 target transformation in a mapping. You can configure multiple Google BigQuery V2 targets to write changed data from a CDC source. You can configure multiple pipelines in a mapping to write changed data from multiple CDC sources to multiple Google BigQuery V2 targets.

When the mapping task processes the changed data from a CDC source such as Oracle Express CDC V2, Google BigQuery V2 Connector creates a state table and a staging table in Google BigQuery. When the changed data is received from the CDC source, Google BigQuery V2 Connector uploads the changed data to the staging table. Then, it generates a `Job_Id` and writes the `Job_Id` to the state table along with the restart information. Google BigQuery V2 Connector then merges the stage table with the actual target table in Google BigQuery.

Each time you run the mapping task, Google BigQuery V2 Connector creates the state table, if it does not exist, to store the state information. Google BigQuery V2 Connector uses the following naming convention for the state table name:

```
state_table_cdc_<MappingTaskID>_<UniqueIdentifierForTargetInstance(s)>
```

Similarly, Google BigQuery V2 Connector uses the following naming convention for the staging table name:

```
staging_table_cdc_<MappingTaskID>_<TargetInstanceName>
```

Mapping tasks with CDC sources example

Your organization needs to replicate real-time changed data from a mission-critical production system to minimize intrusive, non-critical work, such as offline reporting or analytical operations system. You can use Google BigQuery V2 Connector to capture changed data from any CDC source and write the changed data to

a Google BigQuery target. Add the CDC sources in mappings, and then run the associated mapping tasks to write the changed data to the target.

1. In Data Integration, click **New** > **Mapping** > **Create**.

   The **New Mapping** dialog box appears.

2. Enter a name and description for the mapping.

3. On the Source transformation, specify a name and description in the general properties.

4. On the **Source** tab, select any configured CDC connection and specify the required source properties.

5. On the Target transformation, specify a name and description in the general properties.

6. On the **Target** tab, perform the following steps to configure the target properties:

   a. In the **Connection** field, select the Google BigQuery V2 connection.

   b. In the **Target Type** field, select the type of the target object.

   c. In the **Object** field, select the required target object.

   d. In the **Operation** field, select **Data Driven** to properly handle insert, update, and delete records from the source.

   e. In the **Data Driven Condition** field, leave the field empty.

   f. In the **Update Column** field, select the key columns to upsert or update data to or delete data from Google BigQuery.

   g. In the **Advanced Properties** section, you must select CDC in the **Write Mode** property.

   h. You can only configure the following advanced target properties for CDC mode:

      - Target Dataset ID
      - Target Table Name
      - Job Poll Interval in Seconds
      - Pre SQL
      - Pre SQL Configuration
      - Post SQL
      - Post SQL Configuration

7. On the **Field Mapping** tab, map the incoming fields to the target fields. You can manually map an incoming field to a target field or automatically map fields based on the field names.

8. In the **Actions** menu, click **New Mapping Task**.

   The **New Mapping Task** page appears.

9. In the **Definition** tab, enter the task name and select the configured mapping.

10. In the **CDC Runtime** tab, specify the required properties for the selected CDC source.

    For more information about the **CDC Runtime** properties, see the source properties for the selected CDC source.

11. In the **Schedule** tab, add the following properties in the **Advanced Session Properties** section:

    a. In the **Commit on End of File** field, select the value of the property as **No**.

    b. In the **Recovery Strategy** field, select the value of the property as **Resume from last checkpoint**.

12. Click **Save** > **Run** the mapping task.

    Alternatively, you can create a schedule that runs the mapping task on a recurring basis without manual intervention. You can define the schedule to minimize the time between mapping task runs.

In **Monitor**, you can monitor the status of the logs after you run the task.

Rules and guidelines for Google BigQuery V2 CDC target

Consider the following guidelines when working with a Google BigQuery V2 change data capture (CDC) target:

- Informatica recommends that the Secure Agent, the CDC source, and PowerExchange for CDC are configured in the same region as Google BigQuery.

- To increase performance and avoid run-time environment memory issues, increase the Java heap size in the JVM option for type DTM. Set JVMOption1 to -Xmx1024m in the **System Configuration Details** section of the Secure Agent and restart the Secure Agent.

- To improve performance, specify a higher commit interval for the **Maximum Rows Per Commit** property on the CDC Runtime page in the mapping task wizard. However, in case of failure, recovery takes more time for a higher commit interval.

- It is recommended to use update queries on the CDC source database only if the Google BigQuery target table is partitioned and clustered.

- You must define a column as required in the Google BigQuery target table.

- If you define a column as required in the Google BigQuery target table, you must map a column in the CDC source to the required column in the Google BigQuery target in the mapping.

- When you use a Google BigQuery V2 connection in complex mode, you cannot write changed data from a CDC source to a Google BigQuery V2 target.

- When you capture changed data from a CDC source and the Google BigQuery V2 target contains a repeated column of the Record data type, the mapping fails.

Upsert task operation

When you perform an upsert operation on a Google BigQuery target, you must configure the upsert fields for the target table. You can use an ID field for standard objects. Ensure that you include the upsert field in the field mappings for the task.

Rules and Guidelines

Consider the following rules and guidelines when you perform an upsert operation on a Google BigQuery target without using Merge query:

- You cannot use the streaming mode to write data to a Google BigQuery target.

- When you configure a Google BigQuery V2 connection to use simple or hybrid connection mode, you cannot configure upsert operations for columns of the Record data type and repeated columns.

- When you perform an upsert operation on a Google BigQuery target and if multiple incoming rows attempt to update the same target row, ensure that you select the **Disable Duplicate Update Rows** and the **Enable Merge** target advanced property.

Using Merge query for update, upsert, and delete operations

You can implement the Merge query to perform the following operations on a Google BigQuery target:

- Update
- Upsert
- Delete

To implement Merge query, select the **Enable Merge** option in the advanced target properties.

Rules and Guidelines

Consider the following rules and guidelines when you use Merge query:

- When you configure a Google BigQuery V2 connection to use simple and select CSV as the staging file format, the Google BigQuery target table must not contain columns of record data type.
- When you configure a Google BigQuery V2 connection to use simple, the Google BigQuery target table must not contain repeated columns.
- When you configure a Google BigQuery V2 connection to use hybrid connection mode, the Google BigQuery target table must not contain repeated column as a key field.

Data driven operation for mappings

When you flag rows for an insert, update, delete, or reject operation based on the data driven condition for a Google BigQuery target in a mapping, you must select the **Enable Data Driven** and **Enable Merge** properties.

Rules and Guidelines

Consider the following rules and guidelines when you perform a data driven operation on a Google BigQuery target:

- You cannot parameterize the target connection or object.
- You cannot use the Google BigQuery V2 connection in complex mode.
- Ensure that you do not specify a column of Record data type, repeated columns, or Record data type with repeated columns in the data driven condition.
- When you use a Google BigQuery connection in simple mode, ensure that you do not specify a column of Byte, Record data type, or repeated columns in the data driven condition.
- When you use a Google BigQuery connection in hybrid mode, ensure that you do not specify a column of Byte, Record data type with repeated columns, or Primitive data type with repeated columns in the data driven condition.
- When you define the DD_UPDATE constant in the data driven condition, ensure that there are no null values in the update column.
- When you use the DD_REJECT constant in the data driven condition to reject all the rows, ensure that you have selected the **Persist Staging File After Loading** advanced target property.
- Ensure that the target table is not an external table.
- When you use the Google BigQuery V2 connection in simple connection mode and specify an update column of record data type with nested fields, you must ensure that you select **JSON** as the staging file format.
- You need to specify a condition for a data driven operation. If you keep the condition field empty, the mapping fails.
- You cannot use **Disable Duplicate Update Rows** target advanced property to perform a data driven operation.

Determine the order of processing for multiple targets

You can configure a mapping to write to multiple targets in a single pipeline, with each target configured for any write operation. The order of the target operation is not deterministic.

However, if you want to process the target operations to process in a specific order such as delete, update, and insert, you need to set certain properties in the Secure Agent and in the task properties.

Set -DEnableSingleCommit=true in the Secure Agent properties

Perform the following tasks to set the property for the Secure Agent:

1. Open Administrator and select **Runtime Environments**.
2. Select the Secure Agent for which you want to set the property.
3. On the upper-right corner of the page, click **Edit**.
4. In the **System Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service.
5. Edit the JVM options and set the property to *-DEnableSingleCommit=true*.

| DTM | JVMOption1 | -DEnableSingleCommit=true | ☐ |

Set the EnableSingleCommit property in the task properties

Perform the following tasks to set the property in the task:

1. On the **Schedule** tab in the mapping task properties, navigate to the Advanced Session Properties section.
2. From the **Session Property Name** list, select **Custom Properties**, and set the Session Property Value to **Yes**.

Advanced Session Properties

Session Property Name	Session Property Value
Custom Properties	EnableSingleCommit=Yes

Clustering order

You can create a target table with clustered columns at runtime. Clustering organizes the data based on the specified columns into optimally-sized storage blocks, which enhances the query performance.

The clustering order determines the sort order of the data within the table. To define the clustering order, enter up to four fields, each separated by a comma.

You can use the following data types in the clustering columns:

- Bignumeric
- Bool
- Date
- Datetime
- INT64
- Numeric
- String
- Timestamp

Rules and guidelines

Consider the following rules and guidelines when you configure clustered tables in the Google BigQuery target:

- When you set the clustering order, the mapping can fail for the following issues:
  - The number of clustering fields exceeds the defined limit.

- An incorrect name for the clustering field.

- The clustering field is of the Byte or Float data type.

The related error message for the mapping failure is not logged in the session log. You can find the related error message logged in the Tomcat log.

- When you use Simple connection mode in a mapping, you need to provide the clustering field name for the Record data type in the **Clustering Order** property using an underscore instead of a dot. For example, provide the field name as `_(Master_String)` instead of `.(Master.String)`. If you use a dot in the clustering field name, the mapping fails.

Stop on errors

You can configure the maximum allowable errors that a mapping task can encounter before it fails.

The error threshold can include malformed rows, data conversion errors, and load job errors, but the errors resulting from update, delete, or merge queries are excluded.

You can set the error threshold value in the **Stop on errors** session property in a mapping task. You can set the value from 0 to 2147483647. Values less than 0, alphabets, or special characters are not allowed.

To use the stop on error functionality in a mapping task, ensure that you do not select the **Enable BigQuery Storage API** property in the connection and select Bulk as the write mode, where the data format is CSV or JSON.

The following examples describe the mapping behavior based on the error threshold you set:

- If you don't set a value or set the value to 0, the mapping fails at the first error.
- If you set the value to 3, the mapping runs successfully for the first two errors and then fails when the third error occurs.

The mapping task maintains an independent error count for each transformation and table partition. When the error threshold exceeds in any of the transformations in a mapping, the mapping fails.

You cannot access the **Error Rows File** for the mapping task from the **Results** section.

Google BigQuery V2 lookups in mappings

You can create lookups for objects in a Google BigQuery V2 mapping. You can retrieve data from a Google BigQuery V2 lookup object based on the specified lookup condition.

When you configure a lookup in a Google BigQuery V2 mapping, you select the lookup connection and lookup object. You also define the behavior when a lookup condition returns more than one match.

You can use the = (Equal to) and != (Not equal to) operators in a lookup condition.

You can add the following lookups to a Google BigQuery object when you configure field mappings in a mapping task:

- Connected with cached or uncached
- Unconnected with cached
- Dynamic lookup cache

You can only configure a cached lookup for mappings in advanced mode.

Note: You cannot configure an uncached lookup for a lookup object that uses a Google BigQuery V2 connection in complex connection mode.

The following table describes the Google BigQuery V2 lookup object properties that you can configure in a Lookup transformation:

Property	Description
Connection	Name of the lookup connection.
Source Type	Type of the source object. Select Single Object or Parameter.
Lookup Object	Name of the Google BigQuery lookup object for the mapping.
Multiple Matches	Behavior when the lookup condition returns multiple matches. You can return all rows, any row, the first row, the last row, or an error. You can select from the following options in the lookup object properties to determine the behavior: - Return first row[1] - Return last row[1] - Return any row - Return all rows - Report error Only return any row, return all rows, and report error is applicable for mappings in advanced mode.

[1] Doesn't apply to mappings in advanced mode.

The following table describes the Google BigQuery V2 lookup object advanced properties that you can configure in a Lookup transformation with caching enabled:

Property	Description
Source Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
Source Table Name	Optional. Overrides the Google BigQuery table name that you specified in the Lookup transformation.
Number of Rows to Read	Specifies the number of rows to read from the Google BigQuery source table.
Allow Large Results[1]	Determines whether Google BigQuery V2 Connector creates arbitrarily large result tables to query large source tables. If you select this option, you must specify a destination table to store the query results.
Query Results Table Name[1]	Required if you select the **Allow Large Results** option. Specifies the destination table name to store the query results. If the table is not available in the dataset, Google BigQuery V2 Connector creates the destination table with the name that you specify.
Job Poll Interval in Seconds[1]	The number of seconds after which Google BigQuery V2 Connector polls the status of the read job operation. Default is 10.

Property	Description
Read Mode	Specifies the read mode to read data from the Google BigQuery source. You can select one the following read modes: - Direct. In direct mode, Google BigQuery V2 Connector reads data directly from the Google BigQuery source table. **Note:** When you use hybrid and complex connection mode, you cannot use direct mode to read data from the Google BigQuery source. - Staging[1]. In staging mode, Google BigQuery V2 Connector exports data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery V2 Connector downloads the data from Google Cloud Storage into the local stage file and then reads data from the local stage file. Default is Direct mode.
Number of Threads for Downloading Staging Files[1]	Specifies the number of files that Google BigQuery V2 Connector downloads at a time to enable parallel download. This property applies to staging mode.
Local Stage File Directory[1]	Specifies the directory on your local machine where Google BigQuery V2 Connector stores the Google BigQuery source data temporarily before it reads the data. This property applies to staging mode.
Staging File Name[1]	Name of the staging file where data from the Google BigQuery source table is exported to Google Cloud Storage. This property applies to staging mode.
Data Format of the staging file[1]	Specifies the data format of the staging file. You can select one of the following data formats: - Avro - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - Parquet - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ. This property applies to staging mode.
Enable Staging File Compression[1]	Indicates whether to compress the size of the staging file in Google Cloud Storage before Google BigQuery V2 Connector reads data from the staging file. You can enable staging file compression to reduce cost and transfer time. This property applies to staging mode.
Persist Destination Table[1]	Indicates whether Google BigQuery V2 Connector must persist the query results table after it reads data from the query results table. By default, Google BigQuery V2 Connector deletes the query results table.
pre SQL[1]	SQL statement that you want to run before reading data from the source. For example, if you want to select records in the database before you read the records from the table, specify the following pre SQL statement: `SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;`

pre SQL Configuration[1]	Specify a pre SQL configuration.
	For example,
	`DestinationTable:PRESQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

Property	Description

post SQL[1]	SQL statement that you want to run after reading data from the source.
	
	For example, if you want to update records in a table after you read the records from a source table, specify the following post SQL statement:
	
	`UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	`SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where`
	`fullname='John Doe'`

post SQL Configuration[1]	Specify a post SQL configuration.
	
	For example,
	
	`DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:True,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

SQL Override Query[1]	Overrides the default SQL query used to read data from the Google BigQuery source. When you specify SQL override query, you must specify a dataset name in the Source Dataset ID advanced source property. Ensure that the list of selected columns, data types, and the order of the columns that appear in the query matches the columns, data types, and order in which they appear in the source object. Ensure that you only map all the columns in the SQL override query to the target. Does not apply when you enable partitioning.

Use Legacy SQL for SQL Override[1]	Indicates that the SQL Override query is specified in legacy SQL.
	
	Use the following format to specify a legacy SQL query for the **SQL Override Query** property:
	
	`SELECT <Col1, Col2, Col3> FROM [projectID:datasetID.tableName]`
	
	Clear this option to define a standard SQL override query.
	
	Use the following format to specify a standard SQL query for the **SQL Override Query** property:
	
	`SELECT * FROM 'projectID.datasetID.tableName'`

Retry Options[1]	Comma-separated list to specify the following retry options:
	- Retry Count. The number of retry attempts to read data from Google BigQuery.
	- Retry Interval. The time in seconds to wait between each retry attempt.
	- Retry Exceptions. The list of exceptions separated by pipe (|) character for which the retries are made.
	
	Use the following format to specify the retry options:
	
	For example,
	
	`RetryCount:5,RetryInterval:1,RetryExceptions:java.net.ConnectException|`
	`java.io.IOException`
	
	Note: The retry options are available for preview. Preview functionality is supported for evaluation purposes but is unwarranted and is not production-ready. Informatica recommends that you use in non-production environments only. Informatica intends to include the preview functionality in an upcoming release for production use, but might choose not to in accordance with changing market or technical circumstances. For more information, contact Informatica Global Customer Support. To use the functionality, your organization must have the appropriate licenses.

Number of Spark Partitions[2]	Specifies the maximum number of partitions that the Spark engine splits the data into. Default is 1.
Billing Project ID[1]	The project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs query and extract jobs. If you omit the project ID here, the Secure Agent runs query and extract jobs in the Google Cloud project corresponding to the **Project ID** value specified in the Google BigQuery V2 connection.

Property	Description
Lookup Data Filter[1]	Limits the number of lookups that the mapping performs on the cache of the lookup source table based on the value you specify in the filter condition. This property is applicable when you select object as the source type and enable lookup cache on the **Advanced** tab for the Lookup transformation. Maximum length is 32768 characters. For more information about this property, see *Transformations* in the Data Integration documentation.

[1] Doesn't apply to mappings in advanced mode.
[2] Applies only to mappings in advanced mode.

Unconnected lookup transformation

You can configure an unconnected Lookup transformation for the Google BigQuery source in a mapping. Use the Lookup transformation to retrieve data from Google BigQuery based on a specified lookup condition.

An unconnected Lookup transformation is a Lookup transformation that is not connected to any source, target, or transformation in the pipeline.

An unconnected Lookup transformation receives input values from the result of a :LKP expression in another transformation. The Integration Service queries the lookup source based on the lookup ports and condition in the Lookup transformation and passes the returned value to the port that contains the :LKP expression. The :LKP expression can pass lookup results to an expression in another transformation.

Note: You cannot configure a uncached unconnected Lookup transformation for the Google BigQuery source in a mapping.

For more information about the Lookup transformation, see *Transformations*.

Configuring an unconnected lookup transformation

To configure an unconnected Lookup transformation, select the **Unconnected Lookup** option, add incoming fields, configure the lookup condition, and designate a return value. Then configure a lookup expression in a different transformation.

1. Add a Lookup transformation in a mapping.

2. On the **General** tab of the Lookup transformation, enable the **Unconnected Lookup** option.

3. On the **Incoming Fields** tab of the Lookup transformation, create an incoming field for each argument in the :LKP expression.

   For each lookup condition you create, add an incoming field to the Lookup transformation.

4. In the **Lookup Object** tab, import the lookup object.

The **Multiple Matches** property value **Return all rows** in an unconnected lookup is not supported.

5. Designate a return value.

You can pass multiple input values into a Lookup transformation and return one column of data. Data Integration returns one value from the lookup query. Use the return field to specify the return value.

6. Configure a lookup expression in another transformation.

Provide input values for an unconnected Lookup transformation from a :LKP expression in a transformation that uses an Expression transformation. The arguments are local input fields that match the Lookup transformation input fields used in the lookup condition.

7. Map the fields with the target.

Enabling lookup caching

When you configure a Lookup transformation in a mapping, you can cache the lookup data during the runtime session.

When you select **Lookup Caching Enabled**, Data Integration queries the lookup source once and caches the values for use during the session, which can improve performance. You can specify the directory to store the cached lookup.

Lookup Cache Persistent

> Use lookup cache persistent to save the lookup cache file to reuse it the next time Data Integration processes a Lookup transformation configured to use the cache.
>
> You can specify the file name prefix to use with persistent lookup cache files in the **Cache File Name Prefix** field.
>
> If the lookup table changes occasionally, you can enable the **Re-cache from Lookup Source** property to rebuild the lookup cache.
>
> When you enable lookup cache persistent, the mappings don't show messages related to staging optimization.

Dynamic Lookup Cache

> Use a dynamic lookup cache to keep the lookup cache synchronized with the target. By default, the dynamic lookup cache is disabled and represents static cache.
>
> If the cache is static, the data in the lookup cache does not change as the mapping task runs.
>
> If the task uses the cache multiple times, the task uses the same data. If the cache is dynamic, the task updates the cache based on the actions in the task, so if the task uses the lookup multiple times, downstream transformations can use the updated data.

For information about lookup caching, see *Transformations* in the Data Integration documentation.

Optimize lookup performance in staging mode

You can configure Data Integration to optimize the staging performance of a lookup operation.

You can enhance the lookup operation performance by setting a staging property, `INFA_DTM_LKP_STAGING_ENABLED_CONNECTORS`, for the Secure Agent. Data Integration first copies the data from Google BigQuery source into a flat file located in the local staging file directory. When the staging file contains all the data, Data Integration processes this data read.

Consider the following rules when you enable the staging property:

- You can optimize the staging performance when you use connected and cached lookup operation.

- If you run a mapping enabled for SQL ELT optimization, the mapping does not consider the staging property and runs without staging optimization.

- When you read data of the byte data type from the Google BigQuery source, ensure that the size or precision of the binary data does not exceed 62,914,560 bytes.

- Ensure that the total size or precision of all the columns in the Google BigQuery source does not exceed 125,829,120 bytes.

- If the format of the staging file is CSV and you read from a single Google BigQuery table with multiple objects as the source type, the mapping runs without staging optimization.

- If you do not specify a valid path for the local staging file directory, the mapping fails and the session logs do not display a meaningful error message.

- When you parameterize both the Google BigQuery object type and the advanced fields, and select the **Allow Parameter to be overridden at run time** option while configuring the input parameters, the mapping does not consider the staging property and runs without staging optimization.

Enabling Google BigQuery V2 Connector to optimize the lookup performance

Perform the following steps to set the staging property for the Tomcat in the Secure Agent properties:

1. In Administrator, click **Runtime Environments**. The Runtime Environments page appears.
2. Select the Secure Agent for which you want to set the custom configuration property.
3. Click **Edit Secure Agent** icon corresponding to the Secure Agent you want to edit in Actions. The Edit Secure Agent page appears.
4. In the **System Configuration Details** section, select the **Service** as **Data Integration Server** and the type as **Tomcat**.
5. Set the value of the Tomcat property *INFA_DTM_LKP_STAGING_ENABLED_CONNECTORS* to the plugin ID of the Google BigQuery V2 Connector.
   You can find the plugin ID in the manifest file located in the following directory:

   ```
   <Secure Agent installation directory>/downloads/<GoogleBigQueryV2 package>/CCIManifest
   ```
6. Click **Save**.
7. Restart the Data Integration Service.
8. In the Google BigQuery V2 connection, set the `UseRFC4180CSVParser:true` custom property in the **Provide Optional Properties** connection property.

You can check the session logs. If the flat file is created successfully, Data Integration logs the following message in the session log:

```
The reader is configured to run in [DTM_STAGING_CSV] mode.
```

In the Google BigQuery advanced source properties, set the lookup mode as staging and set the Data Format of the staging file property to CSV.

When you enable the staging mode to lookup source data, you can see the following message in the logs:

```
LKPDP_1:READER_1_1> SDKS_38636 [2023-04-21 15:07:23.020] Plug-in #601601: DTM Staging is
enabled for connector for Source Instance [Source].
```

When you disable the staging mode to lookup source data, you can see the following message in the logs:

```
LKPDP_1:READER_1_1> SDKS_38637 [2023-04-21 15:42:45.538] Plug-in #601601: DTM Staging is
disabled for connector for Source Instance [Source].
```

Rules and guidelines for Lookup transformation

Certain rules and guidelines apply when you configure a Lookup transformation.

When you configure a Google BigQuery lookup, adhere to the following guidelines:

- If you specify an SQL override query in a Lookup transformation and configure the Multiple Matches property to Return first row or Return last row, the mapping fails. If you configure the Multiple Matches property to Return any row, Return all rows, or Report error, the Secure Agent displays an error message.
- When you use a Google BigQuery V2 connection in hybrid mode in a Lookup transformation, you cannot configure a lookup condition for fields of bytes, boolean, record, or repeated data types.
- When you specify a custom query to import a lookup table, you cannot configure an uncached lookup.
- When you use a Google BigQuery V2 connection in simple mode in a Lookup transformation, you cannot configure a lookup condition for fields of numeric, date, or datetime data types.

- You must set the **On Multiple Matches** property to **Report Error** when you use a dynamic lookup cache. To reset the property, change the dynamic lookup to a static lookup, change the property, and then change the static lookup to a dynamic lookup.

- When you configure a lookup transformation to read data from a column DataTime data type from a Google BigQuery source, ensure that the year value is lesser than 2200. Otherwise, the lookup returns null value.

- When you configure a Lookup transformation and you specify a column of Float data type in the lookup condition, the Secure Agent fails to match and return the NaN values.

- If you configure an uncached Lookup transformation to look up data from a Google BigQuery source and the source contains a column of DateTime or Timestamp data type, the mapping fails for certain values with the following error:
  `java.lang.RuntimeException: Invalid expression string for filter condition.`

- If you configure an uncached Lookup transformation to look up data from a Google BigQuery source, ensure that the lookup table name does not begin with a number.

- If you configure an uncached Lookup transformation to look up data from a Google BigQuery source, ensure that the lookup condition does not contain a field of the byte data type. Otherwise, the mapping fails with the following error:
  `[ERROR] com.informatica.cci.cloud.client.impl.CCIClientExceptionImpl: Invalid expression string for filter condition.`

- If you configure an uncached Lookup transformation to look up data from a Google BigQuery source, ensure that the lookup condition does not contain a field of the numeric data type. Otherwise, the mapping fails with the following error:
  `[ERROR] java.lang.NumberFormatException`

- When you configure an uncached lookup transformation to read BigNumeric data from a Google BigQuery source and use a Google BigQuery V2 connection in hybrid mode, ensure that all the following conditions are not true:
  - You have configured staging mode to read data from the lookup object.
  - You have configured CSV or JSON as the data format of the staging file.
  - You have configured an advanced lookup filter on the column of BigNumeric data type.

  Otherwise, the mapping fails.

- When you configure an uncached Lookup transformation, map the lookup ports to the target. Otherwise, the mapping fails with the following error:

  `[ERROR] No ports are mapped from lookup transformation. Please map a port for uncached lookup to work.`

- To use the **Is Null** and **Is Not Null** operators in an uncached lookup, you need to set the `-DENABLE_NULL_FLAG_FOR_UNCACHED_LOOKUP=true` property as a JVM option under the DTM type in the Secure Agent properties.

- To enhance the lookup performance in a cached Lookup transformation, configure the following settings:
  - Set the `-DENABLE_SORTED_INPUT_FOR_LKP=true` property as a JVM option under the DTM type in the Secure Agent properties.
  - Set the **Read Mode** property to Staging in the source advanced properties.
  - Enable the **Sorted Input** and **Use EXPORT DATA Statement to stage** properties in the lookup advanced properties.

Read modes

When you use Google BigQuery V2 Connector, you can read data by using direct mode or staging mode. Before you choose a mode, see the Google documentation to understand the cost implications and trade-offs for each mode.

You can read data from a Google BigQuery source by using one of the following modes:
Direct Mode

Use direct mode when the volume of data that you want to read is small. In direct mode, Google BigQuery V2 Connector directly reads data from a Google BigQuery source. You can configure the number of rows that you want Google BigQuery V2 Connector to read in one request.

Staging Mode

Use staging mode when you want to read large volumes of data in a cost-efficient manner.

In staging mode, Google BigQuery V2 Connector first exports the data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery V2 Connector downloads the data from Google Cloud Storage into a local stage file. You can configure the local stage file directory in the advanced source properties. Google BigQuery V2 Connector then reads the data from the local stage file.

When you enable staging file compression, Google BigQuery V2 Connector compresses the size of the staging file in Google Cloud Storage. Google BigQuery V2 Connector then downloads the staging file and decompresses the staging file before it reads the file. To improve the performance and download data in parallel, you can configure the number of threads for downloading the staging file.

If a job fails, Google BigQuery V2 Connector deletes the staging file unless you configure the task or mapping to persist the staging file.

Optimize read performance in staging mode

You can configure Data Integration to create a flat file for staging when you read data from a Google BigQuery source to optimize the staging performance.

You can enhance the read operation performance by setting a staging property, `INFA_DTM_RDR_STAGING_ENABLED_CONNECTORS`, for the Secure Agent. Data Integration first copies the data from Google BigQuery source into a flat file located in the local staging file directory. When the staging file contains all the data, Data Integration reads the data.

You can optimize the staging performance when you read data from single or multiple Google BigQuery objects.

Enabling Google BigQuery V2 Connector to optimize the read performance

Perform the following tasks to set the staging property:

1. In Administrator, click **Runtime Environments**. The Runtime Environments page appears.

2. Select the Secure Agent for which you want to set the custom configuration property.

3. Click **Edit Secure Agent** icon corresponding to the Secure Agent you want to edit in Actions. The Edit Secure Agent page appears.

4. In the **System Configuration Details** section, select the **Service** as **Data Integration Server** and the type as **Tomcat**.

5. Set the value of the Tomcat property `INFA_DTM_RDR_STAGING_ENABLED_CONNECTORS` to the plugin ID of the Google BigQuery V2 Connector.

You can find the plugin ID in the manifest file located in the following directory:

```
<Secure Agent installation directory>/<GoogleBigQueryV2 package>/CCIManifest
```

6. Click **Save**.

7. Restart the Secure Agent.

8. In the Google BigQuery V2 connection, set the `UseRFC4180CSVParser:true` custom property in the **Provide Optional Properties** connection property.

You can check the session logs. If the flat file is created successfully, Data Integration logs the following message in the session log:

```
The reader is configured to run in [DTM_STAGING_CSV] mode.
```

In the Google BigQuery advanced source properties, set the read mode as staging and set the Data Format of the staging file property to CSV.

When you enable the staging mode to read source data, you can see the following message in the logs:

```
READER_1_1_1> SDKS_38636 [2022-07-26 14:59:29.056] Plug-in #601601: DTM Staging is enabled
for connector for Source Instance [Source].
```

When you disable the staging mode to read source data, you can see the following message in the logs:

```
READER_1_1_1> SDKS_38637 [2022-07-26 16:46:04.312] Plug-in #601601: DTM Staging is disabled
for connector for Source Instance [Source].
```

Rules and guidelines when you optimize the read performance

Consider the following rules when you enable the staging property:

- If you run a mapping enabled for SQL ELT optimization, the mapping does not consider the staging property and runs without staging optimization.

- When you read data of the byte data type from the Google BigQuery source, ensure that the size or precision of the binary data does not exceed 62,914,560 bytes.

- Ensure that the total size or precision of all the columns in the Google BigQuery source does not exceed 125,829,120 bytes.

- If the format of the staging file is CSV and you read from a single Google BigQuery table with multiple objects as the source type, the mapping runs without staging optimization.

- If you do not specify a valid path for the local staging file directory, the mapping fails and the session logs do not display a meaningful error message.

- When you parameterize both the Google BigQuery object type and the advanced fields, and select the **Allow Parameter to be overridden at run time** option while configuring the input parameters, the mapping does not consider the staging property and runs without staging optimization.

- When you configure staging optimization to process source data with the Numeric data type with a scale greater than 9 and a precision greater than 28, the mapping truncates the data to a scale of 9 while writing to the target. To preserve the same scale and precision of the Numeric data type in the target, perform the following tasks:

  - Map the BigNumeric data type in the source to the Decimal data type in the target.

  - Create a table in the backend database with a scale of 9.

  - Specify the rounding-off mode for the data.

- When you configure staging optimization to process data and the source contains data of the Numeric or BigNumeric data types with a precision greater than 28 digits, the mapping fails with the following error:
  ```
  [ERROR] Error occurred for Transformation - Source while writing the data to DTM Buffer -
  Data Conversion Failed
  ```

Write modes

When you use Google BigQuery V2 Connector, you can write data by using bulk mode or streaming mode. Before you choose a mode, see the Google documentation to understand the cost implications and trade-offs for each mode.

You can write data to a Google BigQuery target by using one of the following modes:

Bulk mode

Use bulk mode when you want to write large volumes of data in a cost-efficient manner.

In bulk mode, Google BigQuery V2 Connector first writes the data to a staging file in Google Cloud Storage. When the staging file contains all the data, Google BigQuery V2 Connector loads the data from the staging file to the Google BigQuery target.

When you enable staging file compression, Google BigQuery V2 Connector compresses the size of the staging file before it writes data to Google Cloud Storage. Google BigQuery V2 Connector writes the compressed file to Google Cloud Storage and then submits a load job to the Google BigQuery target.

Note: Enabling compression reduces the time that Google BigQuery V2 Connector takes to write data to Google Cloud Storage. However, there will be a performance degradation when Google BigQuery V2 Connector writes data from Google Cloud Storage to the Google BigQuery target.

After writing the data into the Google BigQuery target, Google BigQuery V2 Connector deletes the staging file unless you configure the task or mapping to persist the staging file. You can choose to persist the staging file if you want to archive the data for future reference.

If a job fails, Google BigQuery V2 Connector deletes the staging file unless you configure the task or mapping to persist the staging file.

Streaming mode

Use streaming mode when you want the Google BigQuery target data to be immediately available for querying and real-time analysis. Evaluate Google's streaming quota policies and billing policies before you use streaming mode.

In streaming mode, Google BigQuery V2 Connector directly writes data to the Google BigQuery target. Google BigQuery V2 Connector appends the data into the Google BigQuery target.

You can configure the number of rows that you want Google BigQuery V2 Connector to stream in one request. If you want to stream a larger number of rows than the maximum permissible limit prescribed by Google, you can write the data to multiple smaller target tables instead of one large target table. You can create a template table based on which Google BigQuery must create multiple tables. You can define a unique suffix for each table. Google BigQuery creates each table based on the template table and adds the suffix to uniquely identify each table.

CDC mode

Use CDC mode only when you capture changed data from a CDC source. In CDC mode, you can configure Google BigQuery V2 Connector to capture changed data from any CDC source and write the changed data to a Google BigQuery target table.

Optimize write performance in staging mode

You can configure Data Integration to create a flat file for staging when you write data to a Google BigQuery target in bulk mode. You can set Data Integration to optimize the staging performance.

Data Integration first writes the data to a flat file located in the local staging file directory. When the staging file contains all the data, Data Integration loads the data from the staging file to the Google BigQuery target.

In the Google BigQuery advanced target properties, set the **Local Stage File Directory** property to a directory on your local machine where you want to create the flat file and set the **Data Format of the staging file** property to **CSV**.

When you run the mapping, the flat file is created in the local stage file directory that you specified.

Note: If you do not specify a local stage file directory, the flat file is created in the temp directory in the Linux or Windows machine where the Secure Agent runs.

When the mapping run is completed, the Secure Agent deletes the local staging file.

Enabling Google BigQuery V2 Connector to optimize the staging performance

Perform the following tasks to set the staging property, `INFA_DTM_STAGING_ENABLED_CONNECTORS`, for the Tomcat in the Secure Agent properties:

1. In Administrator, click **Runtime Environments**.
   The Runtime Environments page appears.

2. Select the Secure Agent for which you want to set the custom configuration property.

3. Click **Edit Secure Agent** icon corresponding to the Secure Agent you want to edit in Actions .The Edit Secure Agent page appears.

4. In the **System Configuration Details** section, select the **Service** as **Data Integration Server** and the type as **Tomcat**.

5. Set the value of the Tomcat property *INFA_DTM_STAGING_ENABLED_CONNECTORS* to the plugin ID of the Google BigQuery V2 Connector.
   You can find the plugin ID in the manifest file located in the following directory:

   `<Secure Agent installation directory>/downloads/<GoogleBigQueryV2 package>/CCIManifest`

   The following image shows the INFA_DTM_STAGING_ENABLED_CONNECTORS property set for the Secure Agent:

Tomcat	INFA_DTM_STAGING_ENABLED_CONNECTORS	'601601'

6. Click **Save**.

7. Restart the Secure Agent.

You can check the session logs. If the flat file is created successfully, Data Integration logs the following message in the session log: `INFA_DTM_STAGING mode is enabled for the write operation`.

If you do not set the staging property, Data Integration performs staging without the optimized settings, which might impact the performance of the task.

Rules and guidelines when you optimize the write performance

Consider the following rules when you enable the staging property:

- If you run a mapping enabled for SQL ELT optimization, the mapping runs without SQL ELT optimization.

- When the mapping writes a column of the String data type that contains null values to a column of the String data type set to Required constraint in the Google BigQuery target table, the job fails and does not write the data to any of the target columns.

- When you write Numeric data to the Google BigQuery target, the Numeric data in the local staging flat file contains trailing zeroes. However, the Secure Agent writes the Numeric data correctly in the Google BigQuery target table.

- When you write data of the Binary data type to the Google BigQuery target, ensure that size or precision of the Binary data does not exceed more than 78643200 bytes.

- When you write data with a precision of more than 15 digits in the float data type, the data becomes corrupted.

Process SQL queries using an SQL transformation

You can configure an SQL transformation to process SQL queries and stored procedures midstream in a Google BigQuery V2 mapping.

When you add an SQL transformation to the mapping, on the SQL tab, you define the database connection and the type of SQL that the transformation processes.

When you call a stored procedure in an SQL transformation, you can use a connected SQL transformation. The transformation is connected to the mapping pipeline.

The SQL transformation can process the following types of SQL statements:

Stored procedure

You can configure an SQL transformation to call a stored procedure in Google BigQuery. The stored procedure must exist in the Google BigQuery database before you create the SQL transformation. When the SQL transformation processes a stored procedure, it passes input parameters to the stored procedure. The stored procedure passes the return value to the output fields of the transformation. You can also configure the SQL transformation to generate one output row for each input row.
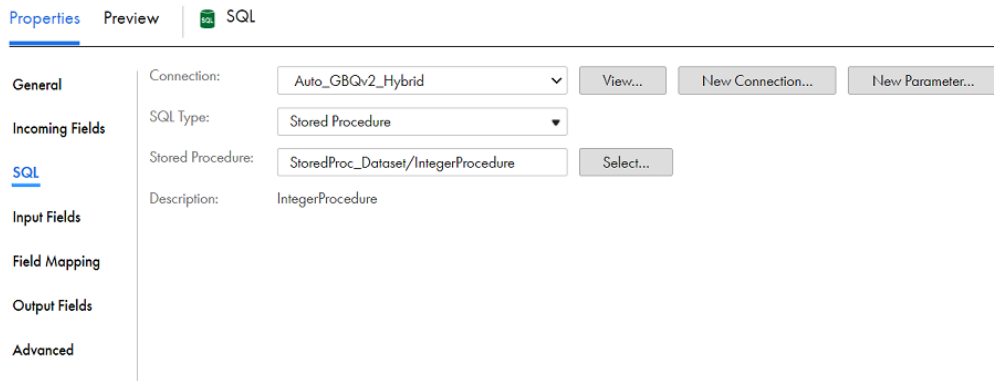
SQL Query

You must use a standard SQL to define the entered SQL query. The SQL transformation processes the query and returns the rows. The SQL transformation also returns any errors that occur from the underlying database or if there is an error in the user syntax to the SQLError output field.
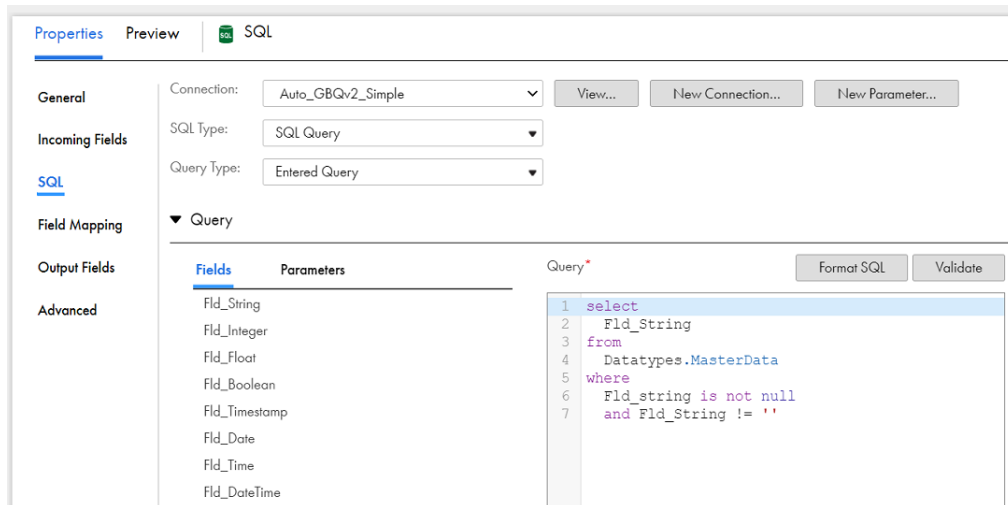
Define multiple entered queries in an SQL transformation separated by a semicolon (;).

Note: Saved query type is not applicable.

For more information about SQL queries and stored procedures, see *Transformations* in the Data Integration help.

When you specify an SQL query in a mapping, you can set the billing project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs the query jobs. If you omit the project ID here, the Secure Agent runs the query jobs in the Google Cloud project corresponding to the **Project ID** value specified in the Google BigQuery V2 connection.

When you configure an SQL query in a mapping, you can assign a label to the SQL transformation. A label is a key-value pair that you can assign to a Google BigQuery resource. Use labels to organize and filter the associated jobs in the Google Cloud Platform Log Explorer.

For more information about labels and their usage requirements, see .

Configuring an SQL transformation

You can configure a SQL transformation to process a stored procedure on the **SQL** tab of the SQL transformation.

This example lists the tasks required to configure an SQL transformation that calls a stored procedure in Google BigQuery.
Your mapping includes user IDs in the data flow. You want to include user names in addition to user IDs. You have a stored procedure that matches user IDs with user names in the database. You add an SQL transformation to your mapping, select the stored procedure, and map the userId incoming field with the userId input field in the stored procedure. Add a SQL transformation in a Google BigQuery mapping.

You check the **Output Fields** tab for the SQL transformation to confirm that it includes the username field. When you run the mapping, the username value is returned with the user ID.
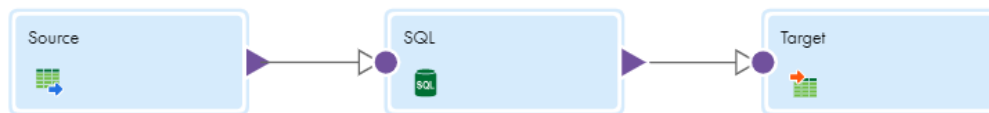
Perform the following tasks in the SQL transformation:

1. Enter a name and description for the SQL transformation.
2. In the **Incoming Fields** tab, define field rules that determine the data to include in the transformation.
3. In the **Properties** panel of the SQL transformation, click the **SQL** tab.
4. In the **SQL** tab, perform the following tasks:
   a. Select the connection to the database.

      You can select the connection or use a parameter.
   b. Set the SQL type to **Stored Procedure**.

c. Click **Select** to select the stored procedure from the database, or enter the exact name of the stored procedure to call.

The stored procedure name is case-sensitive.

Note: If you add a new stored procedure to the database while you have the mapping open, the new stored procedure does not appear in the list of available stored procedures. To refresh the list, close and reopen the mapping.

The following image shows the configured SQL transformation properties:
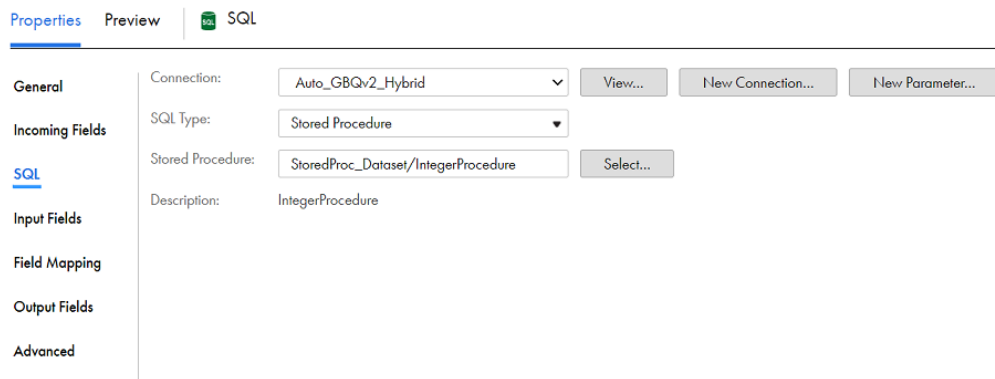
5. In the **Field Mapping** tab, specify how to map incoming fields to the input fields of the selected stored procedure.

6. Define advanced properties for the transformation according to your requirement.

To configure an SQL transformation using the SQL type as SQL entered query , see *Transformations* in the Data Integration help.

Using a parameterized connection in an SQL transformation

You can parameterize a Google BigQuery V2 connection in an SQL transformation.

To parameterize an entered SQL query or a stored procedure in an SQL transformation midstream in a Google BigQuery V2 mapping, perform the following steps:

1. On the **SQL** tab, select a Google BigQuery V2 connection without a parameter.

2. Select **Stored Procedure** or **SQL Query** as the **SQL Type**.

3. Select a stored procedure in Google BigQuery or define an entered query in the SQL editor.

4. Change the connection for the SQL transformation to a parameterized Google BigQuery V2 connection.

Rules and guidelines for SQL transformation

Consider the following rules and guidelines when you configure an SQL transformation in mappings:

- You can only configure a connected SQL transformation with Google BigQuery V2 Connector.

- When you configure an SQL transformation and configure an output field, record data type is not applicable for Google BigQuery V2 Connector.

- When you configure a SQL transformation to call stored procedures or entered query, columns of Record data type and repeated columns are not applicable.

- You cannot call a stored procedure from an entered query.

- When you configure an SQL transformation to process an entered query, the following properties are not applicable:
  - Inout and input parameters
  - Auto commit
  - Transformation scope
  - Stop on error
  - Continue on SQL Error within Row
- When you provide the billing project ID in the SQL transformation advanced properties, you must specify the project ID in the SQL query.

Pre-SQL and post-SQL commands

You can specify **pre SQL** and **post SQL** advanced properties for Google BigQuery sources and targets. When you create a task in Data Integration, you can specify SQL commands in the advanced properties for a source and target.

You can perform the following operations by using pre SQL and post SQL commands:

- SELECT
- UPDATE
- DELETE

Note: You cannot perform more than one operation with a pre SQL or post SQL command.

You can configure the options in Google BigQuery with a pre SQL or post SQL statement in the **pre SQL Configuration** or **post SQL Configuration** advanced properties for Google BigQuery sources and targets.

You must use the following format to specify a pre SQL configuration or a post SQL configuration:

`<Option1:Value1,Option2:Value2,...OptionN:ValueN>`

The following table shows the configuration options and supported values that you can specify in a pre SQL configuration or post SQL configuration:

Options	Supported Values
DestinationDataset	Dataset ID in Google BigQuery
DestinationTable	Table name in Google BigQuery
FlattenResults	True and False
UseLegacySQL	True and False
WriteDisposition	WRITE_TRUNCATE, WRITE_APPEND, and WRITE_EMPTY

Note: If you perform an UPDATE or DELETE operation with a pre SQL or post SQL command, you must specify the following parameter in the pre SQL configuration or post SQL configuration: `UseLegacySQL:False`

Custom query source type

You can use a custom query as a source object when you use a Google BigQuery V2 connection.

You might want to use a custom query as the source when a source object is large. You can use the custom query to reduce the number of fields that enter the data flow. You can configure a custom query to read data from one project while having the Google BigQuery V2 connection set up in another project. You can also create a parameter for the source type when you design your mapping so that you can define the query in the Mapping Task wizard.

To use a custom query as a source, select **Query** as the source type when you configure the source transformation and then use valid and supported SQL to define the query.

You can use legacy SQL or standard SQL to define a custom query. To define a legacy SQL custom query, you must select the **Use Legacy SQL For Custom Query** option when you create a Google BigQuery V2 connection. You can unselect the **Use Legacy SQL For Custom Query** option to define a standard SQL custom query. For more information about Google BigQuery Legacy SQL functions and operators, see Legacy SQL functions and operators.

Rules and guidelines for Google BigQuery custom queries

When you configure a custom query, consider the following guidelines:

- You cannot use custom query as a source for the following configurations:
  - Key range partitions
  - Data filters
  - Sort

- When you specify the SESSSTARTTIME variable in a custom query to return the current date and time, use any of the following formats in the SELECT query:

  - `CAST('$$$SESSSTARTTIME' as TIMESTAMP(0))`

  - `SELECT PARSE_TIMESTAMP('%m/%d/%Y %H:%M:%E6S', '$$$SESSSTARTTIME' ) as t1 --2022-10-06 18:53:28 UTC`

  - `SELECT cast(substr(cast('$$$SESSSTARTTIME' as string),0,19) as datetime FORMAT 'MM/DD/YYYY HH24:MI:SS') as t2;`

- If you provide the billing project ID in the source advanced properties, and you want to use the staging mode for the read operation and enable the **Use EXPORT DATA statement to stage** property, you must specify the project ID in the custom query.

- When you configure a view, you can query only to a maximum of 14 nested levels.

Data filters

You can create simple or advanced data filters. You can also create a set of data filters for each object included in a mapping task. Each set of data filters act independently of the other sets.

You can create simple or advanced data filters for the following data types:

- Integer
- Float
- Numeric (only if you use a Google BigQuery connection in hybrid mode)

- String

- Timestamp

Note: When you create simple or advanced data filters for columns of Numeric data type, ensure that you use a Google BigQuery V2 connection in hybrid connection mode.

You can only configure simple filters for mappings in advanced mode.

Simple data filters

You can create one or more simple data filters. When you create multiple simple data filters, the associated task creates an AND operator between the filters and loads rows that apply to all simple data filters.

Advanced data filters

You can create an advanced data filter to create complex expressions that use AND, OR, or nested conditions. The expression that you enter becomes the WHERE clause in the query used to retrieve records from the source.

Use the following formats to define a filter field in an advanced data filter:

- `<fieldName>`

- `<projectName>.<datasetName>.<tableName>.<fieldName>`

If the Google BigQuery dataset name begins with a number, use the following format to define a filter field in an advanced data filter:

`<projectName>.'<datasetName>'.<tableName>.<fieldName>`

Handling dynamic schemas

You can choose how Data Integration handles changes that you make to the data object schemas. To refresh the schema every time the mapping task runs, you can enable dynamic schema handling in the task.

A schema change includes one or more of the following changes to the data object:

- Fields added, deleted, or renamed.

- Fields updated for data type, precision, or scale.

Configure schema change handling on the **Schedule** page when you configure the task.

The following table describes the schema change handling options:

Option	Description
Asynchronous	Default. Data Integration refreshes the schema when you edit the mapping or mapping task, and when Informatica Intelligent Cloud Services is upgraded.
Dynamic	Data Integration refreshes the schema every time the task runs. You can choose from the following options to refresh the schema: - **Alter and apply changes.** Data Integration applies the following changes from the source schema to the target schema: - New fields: Alters the target schema and adds the new fields from the source. - Renamed fields: Adds renamed fields as new columns in the target. - Data type and precision updates. Not applicable for Google BigQuery V2 Connector. - Deleted fields: Not applicable for Google BigQuery V2 Connector. - **Don't apply DDL changes.** Data Integration does not apply the schema changes to the target. - **Drop current and recreate.** Drops the existing target table and then recreates the target table at runtime using all the incoming metadata fields from the source.

For more information, see the "Schema change handling" topic in *Tasks* in the Data Integration help.

Rules and guidelines for dynamic schema handling

Consider the following rules and guidelines when you enable dynamic schema change handling for mappings:

- When you select the **Alter and apply changes** option on a Google BigQuery table, ensure that the table does not contain a column of record data type or repeated column.

- Schema updates that involve a decrease in the precision of the Varchar data type is not supported.

Configure unique staging object names for concurrent mappings

When you run concurrent mappings to read data from or write data to Google BigQuery using staging mode, you can configure the mapping to create staging objects with unique names.

Google BigQuery V2 Connector creates the following staging objects based on how you configure the mapping:

- Staging tables
- Staging files
- Staging views

To create staging objects with unique names for concurrent mappings, configure the following property in the Google BigQuery V2 connection:

Connection Property	Value
Provide Optional Properties	RandomizeStagingObjectNames:true

Google BigQuery V2 Connector uses the following naming conventions to create unique names for the staging objects based on how you configure the mapping:

Mapping Configuration	Staging Object Type	Staging Object Name Format
Custom Query and SQL Override	Temporary staging view for metadata import	<Informatica_Prefix>_temp_view_for_metadata_<TimestampUptoMilliseconds>_<UUID>
Custom Query and SQL Override	Staging view for runtime	<Informatica_Prefix>_View_<TimestampUptoMilliseconds>_<UUID>
Staging Read Mode	Staging table	<Informatica_Prefix>_<SourceTableName>_<PartitionID[1]>_E_<TimestampUptoMilliseconds>_<UUID>
Staging Read Mode with Custom Query and SQL Override	Staging view	<Informatica_Prefix>_View_Table_Name_<PartitionID[1]>_E_<TimestampUptoMilliseconds>_<UUID>
Staging Read Mode with Staging File Name for CSV, JSON, Avro, or Parquet format	Staging File	<StagingFileName>_<PartitionID[1]>
Staging Read Mode with CSV, JSON, Avro, or Parquet format without specifying the Staging File Name property	Staging File	<Informatica_Prefix>_StgF_<TimestampUptoMilliseconds>_<UUID><PartitionID[1]>.<FileExtension>
Update, Upsert, Delete, or Data Driven operations on a Google BigQuery target	Staging table	<Informatica_Prefix>_<TargetTableName>_<PartitionID[1]>_T_<TimestampUptoMilliseconds>_<UUID>
Lookup	Staging table	<Informatica_Prefix>_LKP_<SourceTableName>__E_<TimestampUptoMilliseconds>_<UUID>_<TableIDIncrement>
Lookup with CSV, JSON, Avro, or Parquet staging file format without specifying the Staging File Name property	Staging file	<Informatica_Prefix>_StgF_<TimestampUptoMilliseconds>_<UUID><StagingFileIncrement>.<FileExtension>
SQL ELT optimization with Google Cloud Storage source and Google BigQuery target	Staging table	<Informatica_Prefix>_<TargetTableName>_PDO_<TimestampUptoMilliseconds>_<UUID>
SQL ELT optimization with Google BigQuery source and target	Staging table	<Informatica_Prefix>_<TargetTableName>_PDO_<TimestampUptoMilliseconds>_<UUID>

Mapping Configuration	Staging Object Type	Staging Object Name Format
SQL ELT optimization with Google BigQuery source and target	Temporary views for custom query and SQL override	<Informatica_Prefix>_View_PDO_ <TimestampUptoMilliseconds>_<UUID>
[1]Applies only when you configure partitions. UUID represents an Unique Universal Identifier string.		

If you select the **Persist Extract Staging File After Download** source advanced property or the **Persist Staging File After Loading** target advanced property, Google BigQuery V2 Connector appends "P" to the end of the <Informatica_Prefix>.

For example when you use Staging Read Mode and select the **Persist Extract Staging File After Download** source advanced property, Google BigQuery V2 Connector uses the following format for the staging table name:

```
<Informatica_Prefix>P_<SourceTableName>_<PartitionID¹>_E_<TimestampUptoMilliseconds>_<UUID>
```

If you cancel a job, Google BigQuery V2 Connector deletes the staging object for the following mapping configurations:

- Staging or direct read mode
- Bulk write mode
- Staging read mode and bulk write Mode with staging file name for CSV, JSON, Avro, or Parquet format
- Custom query
- SQL override
- Update, upsert, delete, or data driven operations on a Google BigQuery target

Assign label to the Google BigQuery resources

A label is a key-value pair that you can assign to a Google BigQuery resource. The labels are associated with load, extract, and query jobs, and you can attach a label for each transformation. You can use labels to organize and filter the associated jobs in the Google Cloud Platform Log Explorer.

A key can have a minimum length of 1 character and a maximum length of 63 characters and cannot be empty. A value can be empty and have a maximum length of 63 characters.

Keys and values can contain only lowercase letters, numeric characters, underscores, and dashes. You need to use characters that are UTF-8 encoded. Also, you can use international characters. The key needs to start with a lowercase letter or an international character. If the label name includes any other characters, the mapping fails.

When you create a transformation in a mapping, the label is optional.

Certain guidelines apply when you add labels for the API calls in a mapping:

- You can apply a maximum of 64 labels in a mapping. However, there are no restrictions on the number of labels that you can use across the mappings within a project.
- You can apply only one label for each transformation. If a mapping contains multiple transformations, you need to provide the label for each transformation.

- You can parameterize the label. To use the current task name as the label, set the label to `$CurrentTaskName`.

- You cannot reuse the key portion of the label within a mapping. However, you can use the same key across mappings.

- In a mapping enabled with source SQL ELT optimization and multiple sources, the label is assigned to all sources. However, the label value from the first source is only used.

Hierarchy Parser transformation in mappings

To preserve the hierarchical structure when you read data from Google BigQuery and write data to relational targets, you must use a Hierarchy Parser transformation.

The transformation processes JSON input from the source transformation and provides relational output to the target transformation. The Hierarchy Parser transformation converts hierarchical input based on the sample schema of the Google BigQuery table that you associate with the transformation and the way that you map the data.

Hierarchy Builder transformation in mappings

When you read data from relational sources and write data to a Google BigQuery target, you must use a Hierarchy Builder transformation.

The transformation processes relational input from the upstream transformation and provides JSON output to the downstream transformation. The Hierarchy Builder transformation produces JSON output based on the sample schema of the Google BigQuery table that you associate with the transformation and the way that you map the data.

Rules and guidelines for mapping and mapping tasks

Certain rules and guidelines apply when you configure a mapping and mapping tasks.

When you configure a Google BigQuery source or Google BigQuery target, adhere to the following guidelines:

- When you create a target at runtime, and the field name in the mapping contains special characters or spaces, the mapping fails.

- When you read from Amazon S3 or Google Cloud Storage and use the update target operation to write to Google BigQuery, ensure that all the columns specified in the update columns field are mapped to the target.

- When you write large datasets to a Google BigQuery target, increase the Java heap size in the JVM options for type DTM. Set JVMOption3 to -Xms1024m and JVMOption4 to -Xmx4096m in the **System Configuration Details** section of the Secure Agent and restart the Secure Agent.

- When you use the Hosted Agent as the runtime environment in a mapping task and use a Hierarchy Builder or Hierarchy Parser transformation in a mapping, you must specify a storage path in Google Cloud

Storage in the **Schema Definition File Path** field under the connection properties. You can then download the sample schema definition file for the Google BigQuery table from the specified storage path in Google Cloud Storage to a local machine.

- Ensure that there is no blank space before you specify the staging directory for the **Local Stage File Directory** property for Google BigQuery source or target. Otherwise, the mapping fails.
- If a Google BigQuery source contains the DATE, DATETIME, TIME, or TIMESTAMP data types and you create a Google BigQuery target at run time, the Secure Agent writes TIMESTAMP data to the target.
- When you read JSON data from a MongoDB source table and write data to a column of Record data type in a Google BigQuery target table, you must specify a explicit value for columns that contain `_id` in the column name. Otherwise, the task fails with the following error:
  ```
  [ERROR] The [LOAD] job failed with the error - [JSON parsing error in row starting at
  position 0:
  ```
- When you use a Google BigQuery V2 connection in simple mode and enable the **Use Legacy SQL For SQL Override** advanced source property, you can only map a single field of repeated data type.
- When you use a Google BigQuery V2 connection in simple mode and configure an advanced data filter condition, ensure that you specify only the column name for the WHERE clause.
  For example, use the following format for the WHERE clause:
  ```
  SELECT <col1>, <col2> FROM `<projectID>.<datasetID>.<tableName>` WHERE <col2>='<value>'
  ```
- When you use a Google BigQuery V2 connection in hybrid mode or complex mode, you must not specify a legacy SQL query for the **SQL Override Query** property. You must clear the **Use Legacy SQL For SQL Override** advanced source property. Otherwise, the mapping fails.
- When you use a Google BigQuery V2 connection in simple mode and enable the **Use Legacy SQL For Custom Query** connection property to define a custom query to read data from a Google BigQuery materialized view, the Secure Agent fails to read the materialized view.
- When you use a Google BigQuery V2 connection in simple mode and read data from a Google BigQuery materialized view as a single source object, you must clear the **Use Legacy SQL For Custom Query** option in the Google BigQuery V2 connection. Otherwise, the Secure Agent fails to read the materialized view.
- When you specify a custom query to read data from Google BigQuery and the source table contains functions as columns, you must specify the alias name for the function.
- If you specify an SQL override query and configure data filters, the mapping runs successfully but the Secure Agent does not consider the filter conditions.
- When you select a Google BigQuery partitioned table as a source or target in a mapping, you cannot preview data.
- When you select a Google BigQuery partitioned table as a source or target in a mapping, you must select the **Use Legacy SQL For Custom Query** connection property.
- When you use month and time partitioned tables in a mapping in simple mode, you must unselect the **Use Legacy SQL For Custom Query** connection property.
- When you configure the Target transformation with a partitioned table, you cannot use `require_partition_filter=true` clause.
- When you use Create New at Runtime to write data to Google BigQuery, the mapping task creates the physical target based on the fields from the upstream transformation in the initial run. But later if you delete the created target table and re-run the mapping task, the Secure Agent fails to create the target table.
- When you use Create New at Runtime to write data to Google BigQuery and the source column name contains special characters, the task fails to create the target table.

- When you perform an update, upsert, or delete operation on a Google BigQuery target without using Merge query and the dataset name, table name, or both starts with a number, the mapping fails. To run the mapping successfully, set the `AllowQuotedIdentifier:true` custom property in the **Provide Optional Properties** connection property.

- When you read null and blank values from a Google BigQuery source in CSV format and perform update operation on a Google BigQuery target, the null values are written as empty strings without quotes and the blank values are written as empty strings with quotes. To treat null values from the Google BigQuery source as null values in the Google BigQuery target, set the `ReadNullAsNullCSVStaging:true` custom property in the **Provide Optional Properties** connection property.

- When you specify a project name, dataset name, or table name in a Google BigQuery V2 mapping, ensure that you do not use reserved keywords.

- When you configure a Google BigQuery target and set the **Data Format of the staging file** to **Avro**, ensure that you do not map fields of DateTime data type.

- When you configure a Google BigQuery target and set the **Data Format of the staging file** to **Parquet**, ensure that you do not map fields of Time and DateTime data type.

- When you configure a Google BigQuery target and provide DestinationTable as any existing table in the **Pre SQL Configuration** and **Post SQL Configuration**, use **Write Disposition** as *Write append* or *Write truncate*. Otherwise, the mapping fails.

- When you run a mapping to read data of timestamp data type from a Google BigQuery source object, incorrect values are written to the target for certain timestamp values.

- When you configure a mapping to read or write data of Record data type with nested fields, ensure that the nested fields do not have the same names.

- When you set the **Data Format of the staging file** to **Parquet** and specify a Google BigQuery dataset name in the **Source Staging Dataset** source advanced property or **Target Staging Dataset** target advanced property to override the **Staging Dataset Name** connection property, ensure that you have the required Google Cloud Storage bucket permission to read data from or write data to Google BigQuery successfully from the staging file in Google Cloud Storage.

- When you configure the **Staging Dataset** connection property and create a new target at runtime, ensure that the Google BigQuery dataset where you want to create the target table has the required permission to create the target successfully.

- When a Google BigQuery source contains columns with the REQUIRED constraint and you use Create New at Runtime to write data, the columns are created with the NULLABLE constraint in the target table.

- When you read data from a column of BigNumeric data type, ensure that you do not select the Avro staging file format. When you write data to a column of BigNumeric data type, ensure that you do not select the Avro or Parquet staging file format.

- When you read data from a column of BigNumeric data type, you cannot specify data filters when you use a Google BigQuery connection in simple or complex mode.

- To pass a column of BigNumeric data type from a Google BigQuery source to a Filter transformation, you must pass the data through an Expression transformation and convert the column to Decimal data type using the TO_DECIMAL() function and map the results to the Filter transformation. Ensure that you specify a precision of 28 and scale of 9.

- When you configure the **Billing Project ID** source advanced properties and if you configure a SQL override, pre-SQL query, or post-SQL query for the Google BigQuery source, you must specify the **Project ID** value specified in the Google BigQuery V2 connection when you define the query.

- When you configure the **Billing Project ID** source advanced properties, you cannot read data from multiple source objects.

- When you configure the **Billing Project ID** target advanced properties, you cannot write data to a Google BigQuery target in CDC mode.

- When you specify the Billing Project ID in the source and target advanced properties and run the mapping, few connector calls appear in the Connection project. However, billing occurs only in the Billing project.

- When you provide an incorrect value for the Billing Project ID property in a mapping, the mapping fails with an irrelevant error message:
```
[ERROR] Error occured while trying to Initialize Data Source Operation |
com.informatica.cci.runtime.internal.utils.impl.CExceptionImpl: !
com.infa.adapter.bigqueryv2.runtime.adapter.BigqueryRuntimeException: Truncate Target
Failed 400 Bad Request

POST https://bigquery.googleapis.com/bigquery/v2/projects/automation-project21/jobs

{"code" : 400, "errors" : [ { "domain" : "global", "message" : "ProjectId and DatasetId
must be non-empty", "reason" : "badRequest" } ], "message" : "ProjectId and DatasetId must
be non-empty", "status" : "INVALID_ARGUMENT" }
```

- When you use $$$SESSSTARTTIME variable in a custom query, the variable returns the session start time as a string value. Use the following syntax to convert the string values to timestamp or datetime:
  - ```
    SELECT PARSE_TIMESTAMP('%m/%d/%Y %H:%M:%E6S', '$$$SESSSTARTTIME' ) as timestamp
    --2022-10-06 18:53:28 UTC
    ```
  - ```
    SELECT cast(substr(cast('$$$SESSSTARTTIME' as string),0,19) as datetime FORMAT 'MM/DD/
    YYYY HH24:MI:SS') as datetime;
    ```
  - ```
    SELECT cast(substr(cast('$$$SESSSTARTTIME' as string),0,19) as timestamp FORMAT 'MM/DD/
    YYYY HH24:MI:SS') as timestamp;
    ```
  When you use SESSSTARTTIME variable in an Expression transformation, the variable returns the session start time as datetime data type.

  The Filter transformation uses system variable as SESSSTARTTIME.

  The Expression transformation uses system variable as SESSSTARTTIME.

- When you use SESSSTARTTIME variable in a custom query without casting and if the source data type for SESSSTARTTIME is string, you might see difference in data format when you compare a mapping that runs with full SQL ELT optimization and a mapping that runs without SQL ELT optimization.
  For example, SESSSTARTTIME returns DateTime value in the MM/DD/YYYY HH24:MI:SS format when a mapping runs with full SQL ELT optimization. When you run the same mapping without SQL ELT optimization, SESSSTARTTIME appends additional zeroes to the return value, MM/DD/YYYY HH24:MI:SS.000000.

- When you run a mapping without SQL ELT optimization, you must use only the columns which are mapped in the Update Column field.

- You must select the **is expression variable** in-out parameter to read a parameter value as an expression in a Filter transformation when you run a mapping without SQL ELT optimization.

- The session log doesn't record the staging optimization error messages in the following scenarios:
  - A mapping with a cached lookup transformation and the staging optimization at source enabled runs without staging optimization.
  - A mapping runs without enabling staging optimization at target.

- When you map a string data type in the source to a time data type in the target, and the data is in the format HH24:MI:SS.US, the mapping fails with the following error:
  ```
  ERROR: Invalid Data Type Conversion
  ```

- When an IN function includes a null value in the list of values in an Expression transformation, the mapping fails with the following error:
  ```
  TE_7002 Transformation stopped due to a fatal error in the mapping. The expression
  [In(col1_string,null,'1','2','3','6',0)] contains the following errors [Function
  validation for [In] failed: [The arguments must be of the same datatype.]. <<PM Parse
  Error>> [In]: : invalid function reference ...
  >>>>In(col1_string,null,'1','2','3','6',0)<<<<].
  ```
  To run the mapping successfully, you can use TO_CHAR(null) function for the string data type and TO_DATE(null) function for the date/time data type.

  The mapping runs successfully if SQL ELT optimization is enabled.

- When you use the IN function in an Expression transformation that includes a column with a null value in the list of values and there is no match, the function returns False instead of NULL.
  If the mapping is enabled with SQL ELT optimization, the function returns NULL.

- If the source field name contains Unicode characters and you use the Create Target option to write to a Google BigQuery target, the mapping fails.

- When you run a mapping to write data of the BigNumeric data type to a new target created at runtime, the scale is not honored in the data. The mapping processes the values up to 28 digits regardless of the scale.

- When you map a BigNumeric data type to the Decimal data type with a precision greater than 28 digits, data truncation occurs at the target.

- When you write data of the BigNumeric data type to a target created at runtime, BigNumeric defaults to Numeric data type in the target. You can use the edit metadata option to change the native data type from Numeric to BigNumeric.

- When you configure an update, delete, or data driven operation to write data and if the metadata in the target does not match the source, the mapping behavior is not deterministic.

- When you configure the `REG_REPLACE(<column_name>, '.* ', <value_to_replace>)` function in a mapping, the function replaces zero or more characters in the `<column_name>` argument with the value provided in the `<value_to_replace>` argument. However, in this scenario, the replaced value is applied twice in the column instead of a single entry when the criteria is met.
  However, a mapping enabled for SQL ELT optimization with the REG_REPLACE() function correctly replaces the value with a single entry.

- When the arguments are null in the REG_REPLACE() function, the mapping fails with the following error:
  ```
  Transformation stopped due to a fatal error in the mapping. The expression
  [Reg_Replace(null,null,null)] contains the following errors [Function validation for
  [Reg_Replace] failed: [The subject and pattern arguments must be of the char datatype.].
  ```
  However, a mapping enabled for SQL ELT optimization with the REG_REPLACE() function runs successfully and returns a null value.

- When you configure a mapping, you cannot use view or materialized view as a target object.

- When you run an existing mapping to write a view as the target object, override the target object with the **Target Table Name** advanced property, and set the optional property `DisableMappingDeployment:true` at the Google BigQuery V2 connection, the mapping fails with the following error:
  ```
  Operation failed: Internal Error Occurred. Contact Support : [Cannot create write
  operation. The node supports read operation only.].
  ```
  To run the mapping successfully, remove the optional property `DisableMappingDeployment:true` at the Google BigQuery V2 connection.

- You can perform the following operations with views in a mapping:

  - Read the data from Google BigQuery views in a Source transformation with staging optimization enabled.

  - Lookup the data from Google BigQuery views in a Lookup transformation.

- You cannot enable staging optimization in a mapping if the **Enable BigQuery Storage API** option is selected in the Google BigQuery V2 connection. A mapping with this configuration fails with the following error:

  ```
  READER_1_1_1> _38644 [2024-02-21 11:56:54.700] Plug-in #601601:Log Message from CCI : [DTM
  staging is not supported when [Enable BigQuery Storage API] is checked].
  ```

- When you configure **Simple** as the connection mode in the Google BigQuery V2 connection and **Staging** as the read mode, you cannot use views as a source or lookup object if the input data contains the Record data type.
  If you run a mapping with this configuration, the mapping fails with the following error:

  ```
  [ERROR] You cannot use [Standard] SQL view as it is not compatible with the Use Legacy SQL
  for Custom Query parameter selection in the Google BigQuery connection. Ensure that the
  view type matches with the selected SQL language in the connection.
  ```

  To run the mapping successfully, you can configure **Hybrid** as the connection mode in the Google BigQuery V2 connection or **Direct** as the read mode.

Rules and guidelines for mappings in advanced mode

Consider the following guidelines when you create a mapping in advanced mode:

- Use a Google BigQuery V2 connection in hybrid mode.

- You cannot configure key range and pass-through partitioning in a mapping that reads from a Google BigQuery source.

- When you perform an update, upsert, or delete operation on a Google BigQuery target table that resides in a region other than the US regions, you must specify the **Region ID** connection property in the Google BigQuery V2 connection.

- When you configure a simple filter condition on a column of Date, Time, Timestamp, or DateTime data type in the source table, ensure that you specify the following DateTime format:
  `YYYY-MM-DD-HH24:MM:SS:MS`

- When you read data from a column of String data type in a Google BigQuery source and write data to a column of Date, DateTime, Time, or Timestamp data type in a Google BigQuery target, ensure that the string data in the source must be of the following DateTime format:
  `YYYY-MM-DD-HH24:MM:SS`

- When you read data from or write data to a partitioned table in Google BigQuery, ensure that you unselect the **Use Legacy SQL for Custom Query** option in the Google BigQuery V2 connection. Otherwise, data preview fails.

- You cannot edit the metadata of the fields of hierarchical data type in a mapping in advanced mode.

- When you import a Google BigQuery source or target table that contains a column of Byte data type, data preview displays blank value for the column of Byte data type.

- When you perform an update operation on a Google BigQuery target, ensure that the update column does not contain NULL values. Otherwise, the Secure Agent fails to update the rows.

- When you perform an upsert operation on a Google BigQuery target, ensure that the update column does not contain NULL values. Otherwise, the Secure Agent performs insert operation instead of update operation.

- When you perform a delete operation on a Google BigQuery target, ensure that the update column does not contain NULL values. Otherwise, the Secure Agent fails to delete the rows.

- When you parameterize the source object, the **Input Parameters** tab in the mapping task displays the source object name as Default.

- When you write data to a Google BigQuery target and specify the staging file name and persist the staging file in Google Cloud Storage, you must delete the staging file after you run the mapping. Otherwise, when you re-run the mapping, the mapping fails with the following error:
  `java.lang.RuntimeException: The object path already exists`

- To achieve maximum throughput when you read data from a large dataset, the value of the **Number of Spark Partitions** must be equal to the number of Spark executors defined for the Google Cloud Platform advanced cluster.

- When you use a parameterized Google BigQuery V2 connection in the Target transformation, the Update Column field does not display in the target properties. In the Target transformation, select a valid connection and object that display in the list and then select the operation.

- When the Google BigQuery target object is parameterized and the selected operation is data driven or upsert in the mapping, the Update Column field does not display in the dynamic mapping task target properties.

- Do not configure an override to the update strategy from the task properties. The agent does not honor the order of precedence and considers the update strategy value specified in the mapping.

- When you read data from and write data to a Google BigQuery column of the Record data type that contains data of the Numeric data type, the precision of the Numeric data must not exceed 15 digits.

- If an existing mapping is set with a precision of 28 digits for the Numeric data type, you can refresh the mapping to set the default precision of 38 digits. However, for hierarchical data types, only up to a precision of 28 digits is applicable.

- When you use Create New at Runtime to write data to Google BigQuery, the mapping task creates the physical target based on the fields from the upstream transformation in the initial run. But later if you delete the created target table and re-run the mapping task, the Secure Agent fails to create the target table.

- You cannot read Integer data in a column of Array or Struct data type from a Google Cloud Storage Avro, JSON, or Parquet file and write the data to a Google BigQuery target.

- You cannot read complex data types that contain multidimensional array from a Google Cloud Storage source and write the data to a Google BigQuery target.

- When you read data from a column of the DateTime data type in a Google BigQuery source, ensure that the column does not contain DateTime values earlier than 1970-01-01. Otherwise, the mapping fails with the following error:
  `java.lang.RuntimeException`

- When you run a mapping to read DateTime data from a Google BigQuery source column of the Record data type with the `YYYY-MM-DD HH24:MI:SS.US` format and write data to a Google BigQuery target column of the Record data type that contains data of the DateTime data type, the Secure Agent truncates the microsecond values and writes the DateTime values in the `YYYY-MM-DD HH24:MI:SS.MS` format.

- When you read data from or write data to a Google BigQuery table, ensure that the Google BigQuery source or target does not contain more that 2000 columns. Otherwise, the mapping fails with the following error:
  `HTTP POST request failed due to IO error.`

- In advanced mode, you cannot write data Null values from an Array data type to a Google BigQuery target. If you do, the mapping fails with a runtime exception.
- The date/time data types in local time zone is converted into UTC when loading data to the Google BigQuery target, resulting in a mismatch in the data between the source and target. Data conversion from String data type to date/time datatypes results in data mismatch for the date/time data types between the source and target.
- When you switch mapping to advanced mode and the mapping uses hierarchical data types, you must manually reimport the Google BigQuery object and create the mapping again.
- When you read or write hierarchical data types in a mapping, the time zone defaults to the Secure Agent host machine time zone. You must change the time zone to UTC time zone to run the mapping successfully. To change to the UTC time zone, you can set the Spark session properties for a specific task from the task properties.
  To set the properties for a specific task, navigate to the Spark session properties in the task properties, and perform the following steps:
  - Select the session property name as `spark.driver.extraJavaOptions` and set the value to `-Duser.timezone=UTC`.
  - Select `spark.executor.extraJavaOptions` and set the value to `-Duser.timezone=UTC`.
- When you configure a mapping in advanced mode that writes the data of more than 10 MB to a row, the mapping fails.
- When you configure a mapping to write Date, Datetime, and Timestamp values, `0001-01-01 00:00:00` or `0001-01-02` into a Timestamp column in the target, the mapping fails. Also, data corruption might occur when you write Date, Datetime, and Timestamp values less than `1910-01-01 00:00:00`.

Troubleshooting a mapping task

Error occurs even though the task runs successfully.

When you parameterize a mapping task and select a Google BigQuery connection in hybrid mode or complex mode, the following error message appears even though the task runs successfully:

```
Internal Error: Adapter InitDataSession failed.
```

Workaround: Ignore the error message.

Job fails when you configure an advanced filter condition.

When you configure an advanced filter condition and run a mapping, the mapping fails with the following error:

```
[ERROR] The [QUERY] job failed with the following error: [Field 'TableName.FldName' not found in table 'DatasetName.TableName'.]
```

Workaround: Remove the table name prefix from the field name in the filter condition and run the mapping.

Job fails when you specify a custom query usng legacy SQL and the field names of a record data type and a primitive data type.

When you configure a Google BigQuery V2 connection in simple mode and enable Use Legacy SQL For Custom Query, the mapping fails if the field names of a record data type and a primitive data type are same.

Workaround: Unselect Use Legacy SQL For Custom Query and run the mapping.

Troubleshooting a mapping in advanced mode

Mapping configured to write Date and Int96 data types for Parquet file fails

A mapping configured to read from Google BigQuery source and write to a Parquet file in Google Cloud Storage target fails in the following cases:

- Data is of the Date data type and the date is less than 1582-10-15.

- Data is of the Int96 data type and the timestamp is less than 1900-01-01T00:00:00Z.

To resolve this issue, specify the following spark session properties in the mapping task or in the custom properties file for the Secure Agent:

- `spark.sql.parquet.int96RebaseModeInWrite=LEGACY`

- `spark.sql.parquet.datetimeRebaseModeInWrite=LEGACY`

- `spark.sql.parquet.int96RebaseModeInRead=LEGACY`

- `spark.sql.parquet.datetimeRebaseModeInRead=LEGACY`

- `spark.sql.avro.datetimeRebaseModeInWrite=LEGACY`

- `spark.sql.avro.datetimeRebaseModeInRead=LEGACY`

Time zone for the Date and Timestamp data type fields defaults to the Secure Agent host machine time zone.

When you run a mapping in advanced mode to read from or write to fields of the Date and Timestamp data types, the time zone defaults to the Secure Agent host machine time zone.

To change the Date and Timestamp to the UTC time zone, you can either set the Spark properties globally in the Secure Agent directory for all the tasks in the organization that use this Secure Agent, or you can set the Spark session properties for a specific task from the task properties:

To set the properties globally, perform the following tasks:

1. Add the following properties to the `<Secure Agent installation directory>/apps/ At_Scale_Server/41.0.2.1/spark/custom.properties` directory:

    - infacco.job.spark.driver.extraJavaOptions=-Duser.timezone=UTC

    - infacco.job.spark.executor.extraJavaOptions=-Duser.timezone=UTC

2. Restart the Secure Agent.

To set the properties for a specific task, navigate to the Spark session properties in the task properties, and perform the following steps:

- Select the session property name as `spark.driver.extraJavaOptions` and set the value to `–Duser.timezone=UTC` .

- Select `spark.executor.extraJavaOptions` and set the value to `–Duser.timezone=UTC`.

CHAPTER 7

Migrating a mapping

You can configure a connection and mapping in one environment and then migrate and run the mapping in another environment.

You can also migrate mappings configured in advanced mode. After the migration, you can change the connection properties from the Administrator service, but you do not need to modify the mapping. Data Integration uses the configured runtime attributes from the earlier environment to run the mapping successfully in the new environment.

Consider a scenario where you develop a mapping in the development organization (Org 1) and you then migrate and run the mapping in the production organization (Org 2). After you migrate, you might want to use the same or a different connection endpoint or object path in Org 2. Based on your requirement, follow the guidelines in this section before you plan the migration.

Use the same object path for the migrated mapping

If you want the migrated mapping in Org 2 to use the same object path as in Org 1, you must maintain the same dataset name and table name in the Google BigQuery account for Org 2.

For example, if you have two different accounts, Account1 used for Org 1 and Account2 used for Org 2, the object path for the dataset name and table name must be the same in both the accounts:

Account1: DatasetName1/TableName1

Account2: DatasetName1/TableName1

In this scenario, you do not need to override the dataset and table name in the advanced properties and the mapping runs successfully.

Use a different object path for the migrated mapping

After you migrate the mapping, you can use a different object path to run the mapping from the new environment.

In this scenario, before you migrate the mapping, you can change the object metadata, runtime attributes, or the connection attributes to reflect the object path in the migrated environment. You do not have to edit or update the mapping in the new environment.

As a rule, when you specify the dataset name and table name in the advanced properties, connection, or object properties, Data Integration honors the attributes in the following order of precedence:

1. **Runtime advanced attributes**. The advanced properties such as dataset name and table name in the Source or Target transformation in a mapping.

2. **Object metadata**. The object selected in the Source or Target transformation in a mapping.

Migration options

When you migrate, you can choose from one of the following options to update the object path:

Option 1. Override the properties from the advanced properties

Before the migration, specify the required dataset and table name for the object from Org 2 in the advanced properties of the Org 1 mapping.

After the migration, when you run the mapping, the Secure Agent uses the configured advanced parameters to override the object specified in the mapping imported from Org 1.

Option 2. Parameterize the properties in the mapping

You can choose to parameterize the advanced attributes, such as the dataset and table name before the migration. You can configure input parameters, in-out parameters, and parameter files in the mapping. When you use a parameter file, you can save the parameter file on a local machine or in a cloud-hosted directory.

After you migrate the mapping, do not edit or update the mapping. If you have used in-out parameters, you can change the dataset and table attributes using the parameter file so that the changes are applied when the task runs.

Parameterizing only the advanced properties, but not the object in the mapping

If you want to parameterize only the advanced properties and use them at runtime, select a placeholder object in the object properties in the mapping and then specify an override to this placeholder object from the advanced properties. Ensure that the placeholder object contains the same metadata as the corresponding table that you specify as an override. When you run the mapping, the value specified in the advanced property overrides the placeholder object.

Parameterizing both the object and the advanced properties

If you want to keep both the Google BigQuery object type and the advanced fields parameterized, you must leave the **Allow parameter to be overridden at runtime** option unselected in the input parameter window while adding the parameters, and then select the required object at the task level. When you run the task, the values specified in the advanced properties take precedence.

Rules and guidelines for migrating a mapping

Consider the following rules and guidelines when you use the same or a different object path for the migrated mapping:

- The following table lists the transformation, object type, and the fields in the advanced properties of a mapping that you can retain when you migrate to the new environment:

Transformations	Object Type	Advanced Fields
Source	Single object, multiple objects	Dataset and table name
Lookup	Single object **Note:** Applicable for unconnected, and connected cached and uncached.	Dataset and table name
Target	Single object	Dataset and table name

- Before you migrate a mapping to the new environment, map the connection from the earlier environment to the migrated environment that has access to the dataset and table name configured in the advanced properties in the earlier environment.

- Ensure that the table that you specify as an override contains the same schema as the corresponding table selected during design time.

- When you specify a custom query as a source object and override the source dataset and source table name, the Secure Agent does not consider the dataset name or table name to import the metadata.

- After you migrate the mapping to Org2, you must not edit the mapping.

- When you override the dataset and table name in a mapping, the mapping fails with an error if the dataset used in the connection in Org2 is not valid. The mapping fails with the following error:
  ```
  Operation failed: error [The following exception occurred: [404 Not Found]
  ```

- Before you migrate a mapping that contains an override to the parameterized object using the parameter file, you need to disable the "Allow parameter to be overridden at run time" option. If you enable the option and you run the mapping after the migration, the following error occurs:
  ```
  Mapping is failing with error -Unable to start Mapping Task because schema fetch failed.
  ```

CHAPTER 8

Upgrading to Google BigQuery V2 Connector

If you are accessing Google BigQuery using the Google BigQuery connection or the Google BigQuery ODBC connection, you can upgrade to the newer Google BigQuery V2 Connector. You can replace the source or target connection type in existing mappings and mapping tasks that use the Google BigQuery connection or the Google BigQuery ODBC connection with the Google BigQuery V2 connection.

After you replace the connection in an existing mapping, reimport the Google BigQuery object and remap the fields in the mapping. The configured advanced source, target, and lookup properties in the fields that are common between the two connectors are retained in the new connector. You can run the mapping successfully using the configured values from the old connector. You can additionally configure features that the enhanced Google BigQuery V2 Connector offers.

Note: If you are using the Google BigQuery ODBC connection or the Google BigQuery connection in mappings to read from or write data to Google BigQuery, Informatica recommends you to use the Google BigQuery V2 connection to make use of the features that the enhanced connector offers. To get the license for Google BigQuery V2 Connector, contact Global Customer support.

Connection switching example

You want to upgrade your existing Google BigQuery V1 mapping that uses the Google BigQuery connection to the Google BigQuery V2 connection.

1. Open the existing Google BigQuery V1 mapping that you want to upgrade to Google BigQuery V2.

   The following image shows an existing mapping that uses the Google BigQuery connection and contains the configured advanced properties in the Target transformation:

   The configured target object in this example is: `MasterData_PreSQLSource`

2. To retain the mapped fields from the field mapping when you switch the connection, on the **Field Mapping** tab, choose from the following **Field Map Options** menu in the Google BigQuery V1 mapping:

   - To retain the fields automatically mapped after the switch, select **Automatic**.

   - To manually map the retained fields after the switch, select **Manual**.

     Note: When you select manual, after switching the connection, you have the option to automap the retained fields using the previous mapping.

3. To switch the connection, in the **Connection** field, change the connection from Google BigQuery V1 to Google BigQuery V2.

4. In the **Change Connection** dialog box, select the following properties, and click **Yes**:

   - **Connection switch**. Switches to the connection that you select.

   - **Retain field mapping**. Retains the configured field mappings from Google BigQuery V1.

The following image shows the options that you must select:
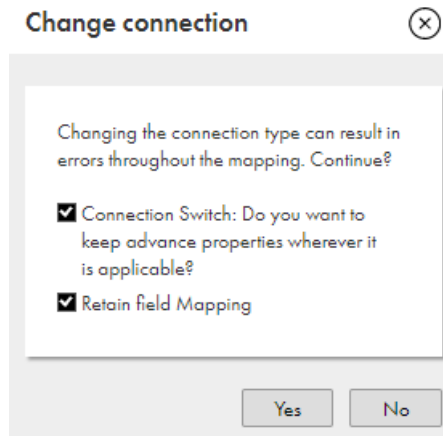
5. Use the same object path in the mapping as Google BigQuery V1.

    The following image shows the switched connection with the same object=: `MasterData_PreSQLSource`
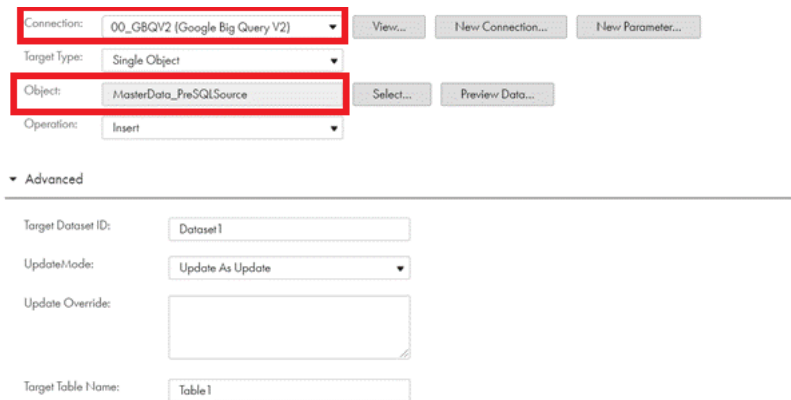
    The configured target advanced properties from the Google BigQuery V1 mapping reflect in the Target transformation.

6. If you had selected **Manual** on the **Field Mapping** tab in the Google BigQuery V1 mapping and you want to reflect the field mappings in Google BigQuery V2, on the **Field Mapping** tab, select **Automap**, and then select **Previous Mapping**.
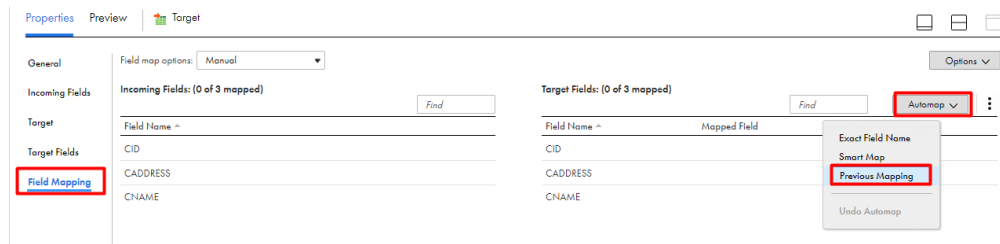
    Note: If you had selected **Automatic** in Google BigQuery V1, you do not have to perform this task.

In the following image, the configured mapped fields from the Google BigQuery V1 mapping reflects in the Google BigQuery V2 mapping:
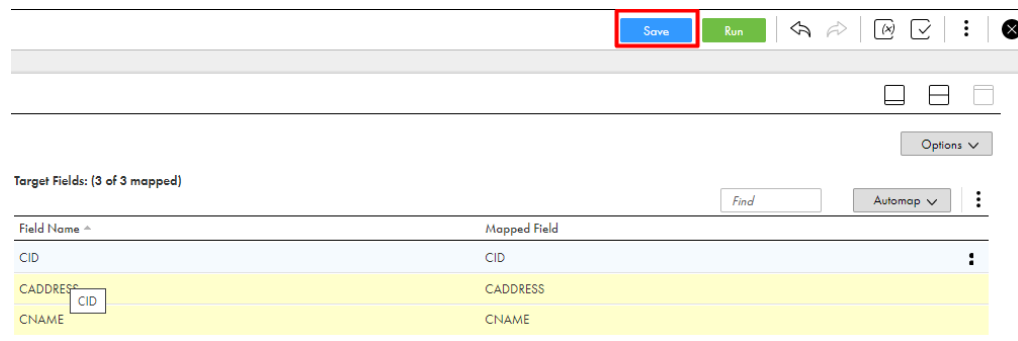
7. Click **Save**.

Advanced properties retained after the switch

The following table lists the configured advanced source and target properties from Google BigQuery ODBC that are retained in Google BigQuery V2 mappings:

Properties	Google BigQuery ODBC properties retained in Google BigQuery V2
Source	- Pre SQL - Post SQL - SQL Override
Target	- Pre SQL - Post SQL - Update Override

The following table lists the configured advanced source, lookup, and target properties from Google BigQuery V1 mappings that are retained in Google BigQuery V2 mappings:

Properties	Google BigQuery V1 properties retained in Google BigQuery V2
Source and Lookup	- Source Dataset ID - Source Table Name - Number of Rows to Read - Allow Large Results - Query Results Table Name - Job Poll Interval In Seconds - Read Mode - Number of Threads for Downloading Staging Files - Data format of the staging file - Local Stage File Directory - Staging File Name - Enable Staging File Compression - Persist Destination Table - pre SQL - post SQL - pre SQL Configuration - post SQL Configuration
Target	- Target Dataset ID - Target Table Name - Create Disposition - Write Disposition - Write Mode - Streaming Template Table Suffix - Rows per Streaming Request - Staging File Name - Data format of the staging file - Persist Staging File After Loading - Enable Staging File Compression - Job Poll Interval In Seconds - Number of Threads for Uploading Staging File - Local Stage File Directory - Allow Quoted Newlines - Field Delimiter - Allow Jagged Rows - pre SQL - post SQL - pre SQL Configuration - post SQL Configuration - Quote Char

Rules and guidelines

Consider the following rules before you replace the Google BigQuery ODBC connection or Google BigQuery V1 connection in an existing mapping with the Google BigQuery V2 connection:

- When you specify the pre-SQL, post-SQL, and SQL query commands in the source and target properties for Google BigQuery and upgrade the connection to Google BigQuery V2, ensure that the number of characters in the queries in the Google BigQuery ODBC connection or Google BigQuery V1 connection must not exceed 65535. Otherwise, the queries are truncated.

- Even after you replace the connection with the Google BigQuery V2 connection, the object field displays the object that you selected in the earlier connection. You need to re-import the Google BigQuery object and remap the fields in the mapping.

CHAPTER 9

Google BigQuery SQL ELT optimization

You can use SQL ELT optimization to push the transformation logic to the Google BigQuery database.

SQL ELT optimization

When you run a task configured for SQL ELT optimization, Data Integration converts the transformation logic to an SQL query. Data Integration sends the query to the database, and the database runs the query. The amount of transformation logic that Data Integration pushes to the database depends on the database, the transformation logic, and the mapping configuration. Data Integration processes all transformation logic that it cannot push to a database.

Configure SQL ELT optimization for a mapping in the tasks properties. You cannot configure SQL ELT optimization for a mapping in advanced mode.

SQL ELT optimization types

When you apply SQL ELT optimization, the task pushes transformation logic to the source or target database based on the optimization type you specify in the task properties. Data Integration translates the transformation logic into SQL queries or Google BigQuery commands to the Google BigQuery database. The database runs the SQL queries or Google BigQuery commands to process the transformations.

You can configure the following types of SQL ELT optimization in a mapping:
Full

Data Integration first pushes as much of the transformation logic as possible to process in the target database. If the target database cannot process some of the transformation logic, it pushes that logic for processing to the source database. Data Integration processes all the remaining transformation logic that it cannot push to the target or source database. This is applicable for mappings that read from or write to Google BigQuery.

When you select full SQL ELT optimization for mappings that read from Google Cloud Storage and write to Google BigQuery, Data Integration pushes as much of the transformation logic as possible to process in the target database. Data Integration processes all the transformation logic that it cannot push to the target database.

Source

Data Integration pushes down as much as the transformation logic as possible to process in the source database.

When you select source SQL ELT optimization, Data Integration pushes the transformation logic for all the supported transformations downstream in the mapping, but excludes the target transformation.

SQL ELT optimization scenarios

You can configure SQL ELT optimization for the following scenarios in mappings:

Important: To configure SQL ELT optimization using the Google BigQuery V2 Connector, verify that your organization has the **Mappings-Advanced SQL ELT Optimization** license. To get the license, contact Global Customer Support.

Source and target endpoints	Supported SQL ELT optimization scenarios in mappings	SQL ELT optimization type
Google BigQuery source Google BigQuery target	Reads from and writes to Google BigQuery using the Google BigQuery V2 connection.	Full, Source
Google Cloud Storage source Google BigQuery target	Reads from Google Cloud Storage using a Google Cloud Storage V2 connection and writes to Google BigQuery using a Google BigQuery V2 connection.	Full
Amazon S3 source Google BigQuery target	Reads from Amazon S3 using an Amazon S3 V2 connection and writes to Google BigQuery using a Google BigQuery V2 connection.	Full
Note: You can use the Secure Agent or the Hosted Agent to run mappings enabled with SQL ELT optimization.		

SQL ELT optimization preview

Before you can run a mapping task configured for SQL ELT optimization, you can preview if SQL ELT optimization is possible when you create the mapping. You can preview SQL ELT optimization from the **SQL ELT Optimization** panel in the Mapping Designer.

After you select the required SQL ELT optimization options and run the preview, Data Integration creates and runs a temporary SQL ELT optimization preview mapping task. When the job completes, Data Integration displays the SQL queries to be executed and any warnings in the **SQL ELT Optimization** panel. The warning messages help you understand which transformations in the configured mapping are not applicable for SQL ELT optimization. If SQL ELT optimization fails, Data Integration lists any queries generated up to the point of failure. You can edit the mapping and fix the required transformations before you run the mapping for SQL ELT optimization.

You can also view the temporary job created under **My Jobs** and download the session log to view the queries generated.

For more information about how to preview SQL ELT optimization, see the topic "SQL ELT optimization preview" in *Mappings* in the Data Integration help.

Configuring SQL ELT optimization

To optimize a mapping, add the mapping to a task, and then configure SQL ELT optimization in the mapping task.

1. Create a mapping task.

2. In the **SQL ELT Optimization** section on the **Runtime Options** tab, set the SQL ELT optimization value to **Full** or **To Source**.

3. If full SQL ELT optimization is not available, select how Data Integration handles SQL ELT optimization in the **SQL ELT Optimization Fallback Option** menu:

    - Partial SQL ELT. Default. Data Integration pushes as much transformation logic as possible to the source database. The task processes any transformation logic that it can't push to a database. You can use Partial SQL ELT only when you read from and write to Google BigQuery.

    - Non SQL ELT. The task runs without SQL ELT optimization.

    - Fail Task. Data Integration fails the task.

    Note: The fallback options are not applicable to mappings in advanced mode.

    When you run the mapping task, the transformation logic is pushed to the Google BigQuery database.

Configuring a custom query or an SQL override for the Google BigQuery source object

You can push down a custom query or an SQL override to Google BigQuery.

Before you run a task that contains a custom query as the source object or you configure an SQL override, you must set the **Create Temporary View** session property in the mapping task properties.

Note: If you do not set the **Create Temporary View** property, the mapping runs without SQL ELT optimization.

Perform the following task to set the property:

1. In the mapping task, navigate to the **SQL ELT Optimization** section on the **Schedule** tab.

2. Select **Create Temporary View**.

3. Click **Finish**.

Context based optimization for multiple targets

When you configure a mapping to write to multiple Google BigQuery targets or write to the same Google target table in two Target transformations, you can further optimize the write operation when you configure full SQL ELT optimization.

To optimize, you can choose to configure an insert, update, upsert, delete, or data driven operation for multiple targets individually. You can select the same Google BigQuery target table in multiple Target transformations and perform different operations for each of the Target transformations to run independent of each other.

When you configure a mapping enabled for full SQL ELT optimization to write to the same Google BigQuery target table in two target transformations, you can specify the optimization context for slowly changing dimension type 2 merge scenario.

You can enable the **SCD Type 2 merge** when you write to same Google BigQuery table in two Target transformations and perform different operations for each of the Target transformations, where you use one target to insert data and the other target to update data. Data Integration combines the queries for both the targets and issues a Merge query.

Note: Multi-insert mode is not applicable for Google BigQuery targets.

Understanding an SCD type 2 merge mapping

The SCD Type 2 merge mapping uses a BigQuery source and two target transformations that write to the same Google BigQuery table. One target transformation updates the table while the other transformation inserts data to the Google BigQuery table.

The following image shows a mapping that writes slowly changing dimension data to a Google BigQuery target table:
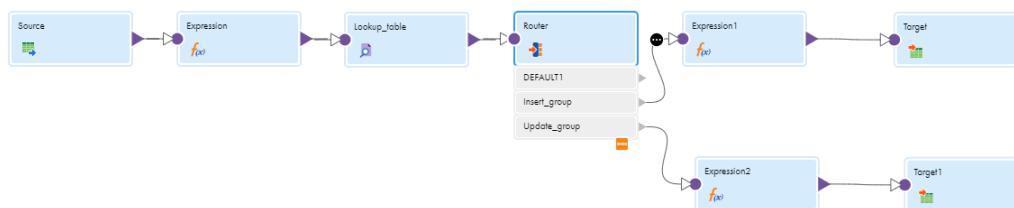
Add expression and lookup transformations to compare source data against the existing target data. You enter the lookup conditions and source columns that you want the Data Integration to compare against the existing target.

For each source row without a matching row in the target, the Expression transformation marks the new row. For each source row with a matching row in the target, the Expression transformation compares existing source and target columns with the MD5() function. If those columns do not match, the Expression marks the existing target row as an inactive row and inserts a new target row as an active row. The mapping then splits into two data groups using the Router transformation.

You must generate an UUID value through the Expression transformation and add it as a unique ID column and also as the first column in the target. Additionally, you can add an active status flag, MD5() hash value, start timestamp, and end timestamp columns to write to the target through the Expression.

The first data flow from the Router transformation passes only new rows to the Expression transformation. The Expression transformation inserts new rows to the target. The Expression transformation also assigns an UUID value and updates the start timestamp, MD5() function hash value, and the active status as 1 for each new row.

In the second data flow, the Router transformation passes only changed rows to pass to the Expression transformation. The Expression transformation inserts changed rows to the target. The Expression transformation updates the active status as 0 and adds the end timestamp for the existing row in the target.

Clean stop a SQL ELT optimization job

When a task enabled for SQL ELT optimization is running, you can clean stop the job to terminate all the issued statements and processes spawned by the job.

You can use this option for mappings enabled for SQL ELT optimization that use the Google BigQuery V2 connection either in the source or target transformation, or both.

Use the **Clean Stop** option on the My Jobs page in Data Integration and the All Jobs and Running Jobs page in Monitor.

See the following exceptions before you clean stop a SQL ELT optimization task:

- When you clean stop a task enabled for source SQL ELT optimization that reads from or writes to Google BigQuery and the target or source properties in the mapping contains pre-SQL or post-SQL statements, even if the select query is terminated, the job continues to run the target post-SQL query.

- When you start a job enabled for full SQL ELT optimization and clean stop it immediately, and if the mapping is configured to create a new target at runtime, the table is created even if the job is terminated.

SQL ELT optimization using a Google BigQuery V2 connection

You can configure SQL ELT optimization for a mapping that contains a Google BigQuery V2 connection. SQL ELT optimization enhances the mapping performance. You can configure full SQL ELT optimization when you read data from an Google Cloud Storage source and write to a Google BigQuery target.

You can configure SQL ELT optimization for a mapping task to read from or write data to Google BigQuery objects associated with different projects in different Google service accounts within the same region.

SQL ELT optimization compatibility

You can configure the task to push transformations, variables, functions, and operators to the database.

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent operators, variables, and functions in the database. If there is no equivalent operator, variable, and function, the Secure Agent processes the transformation logic.

Supported functions

The following table lists the functions that can be pushed to the Google BigQuery database by using full SQL ELT optimization:

Function	Function	Function	Function
ABS()	IN()	MOD()	SYSDATE()
ADD_TO_DATE()	INSTR()	POWER()	SYSTIMESTAMP()
AVG()	IS_DATE()	REG_REPLACE()	TAN()
CEIL()	IS_NUMBER()	REPLACECHR()	TANH()
CHR()	IS_SPACES()	REPLACESTR()	TO_BIGINT
CONCAT()	ISNULL()	ROUND(DATE)	TO_CHAR(DATE)
COS()	LAST_DAY()	ROUND(NUMBER)	TO_CHAR(NUMBER)
COSH()	LENGTH()	RPAD()	TO_CHAR(STRING)
COUNT()	LN()	RTRIM()	TO_DATE()
DATE_COMPARE()	LOG()	SIGN()	TO_DECIMAL()
DATE_DIFF()	LOWER()	SIN()	TO_FLOAT()

Function	Function	Function	Function
DECODE()	LPAD()	SINH()	TO_INTEGER()
EXP()	LTRIM()	SQRT()	TRUNC(DATE)
FLOOR()	MAX()	STDDEV()	TRUNC(NUMBER)
GET_DATE_PART()	MD5()	SUBSTR()	UPPER()
IIF()	MIN()	SUM()	VARIANCE()

Rules and guidelines

When you push functions to Google BigQuery, adhere to the following guidelines:

- To push IS_DATE() function to Google BigQuery, you must configure the output field in the expression transformation to a column of string data type.

- When you push the IS_DATE() function to Google BigQuery and use the SS.US format argument and specify values with the SS.MS or SS format, the IS_DATE() returns true.

- When you push the IS_DATE() function to Google BigQuery and use the MON format argument and specify values with the MONTH format, the IS_DATE() returns true.

- When you push the IS_DATE() function to Google BigQuery and use the MONTH format argument and specify values with the MON format, the IS_DATE() returns true.

- When you use Is_Number(), Is_Spaces(), and Is_Date() in an Expression transformation, the output field type supports only integer and string data types.

- When you push a function to Google BigQuery and the mapping runs without SQL ELT optimization, the IS_DATE() returns Boolean values of 0 or 1 to the Google BigQuery target table. If the mappings run with SQL ELT optimization, the IS_DATE() returns Boolean values of true or false to the Google BigQuery target table.

- When you specify Is_Number(), Is_Spaces(), or Is_Date() functions in an Expression transformation, the output field type supports only integer and string data types.

- To push the TO_CHAR(DATE) function to Google BigQuery, you must use the following string and format arguments:

  - YYYY

  - YY

  - RR

  - Q

  - MM

  - MON

  - MONTH

  - DD

  - DDD

  - DY

  - DAY

  - HH12

- HH24

- MI

- SS

- SS.MS

- SS.US

- am

- AM

- pm

- PM

- To push the TO_DATE(string, format) function to Google BigQuery, you must use the following format arguments:

- YYYY

- YY

- RR

- MM

- MON

- MONTH

- DD

- HH12

- HH24

- MI

- SS

- SS.MS

- SS.US

- am

- AM

- pm

- PM

- To push the ADD_TO_DATE(date, format, amount) or TRUNC(date, format) function to Google BigQuery, you must use the following format arguments:

- YYYY

- YY

- YYY

- Y

- MM

- MON

- MONTH

- D

- DD

- DDD

- DY

- DAY

- HH

- HH12

- HH24

- MI

- SS

- MS

- US

- To push the GET_DATE_PART(date, format) function to Google BigQuery, you must use the following format arguments:

- YYYY

- YY

- YYY

- Y

- MM

- MON

- MONTH

- DD

- DDD

- DY

- DAY

- HH

- HH12

- HH24

- MI

- SS

- MS

- US

- When you push the GET_DATE_PART() function to the Google BigQuery database and specify null in the format argument, the mapping runs without SQL ELT optimization.

- When you push the LAST_DAY() function to the Google BigQuery database and specify Date/Time or Timestamp value in the date argument, the LAST_DAY() function pushes only the date values to the Google BigQuery target. You might encounter a data mismatch when you compare a mapping that runs with full SQL ELT optimization and a mapping that runs without SQL ELT optimization.

- To push the ROUND(DATE) function to Google BigQuery, you must use the following format arguments:

- DD

- DDD

- DY

- DAY

- HH

- HH12

- HH24

- MI

- SS

- MS

- When you push the ROUND(DATE) function to the Google BigQuery database and use NS (nanoseconds) in the format argument, the mapping runs without SQL ELT optimization or with source SQL ELT optimization.

- To push the ROUND(NUMBER) function to Google BigQuery, you must use a numeric value of the following data types:

  - Decimal

  - Numeric

  - NULL

- To push the INSTR() function to Google BigQuery, you must only define the input_field and string arguments.

- When you push the SYSTIMESTAMP() function to the Google BigQuery database, do not specify any format arguments. If you do not specify any format arguments, the Google BigQuery database returns the complete timestamp.

- If you use a % operator in an expression transformation, the mapping converts the % operator to the MOD() function and pushes the MOD() function to Google BigQuery.
  The MOD() function supports arguments of Int64 and Numeric data types. When you push the MOD() function to the Google BigQuery database, ensure that the format arguments are of the same data type. You can specify the arguments in the following formats:

  - MOD(Int64, Int64)

  - MOD(Numeric, Numeric)

- When you push the TRUNC(DATE) function to the Google BigQuery database and specify a NULL value in the format argument, the mapping runs without SQL ELT optimization.

- When you push the SYSDATE() function to the Google BigQuery database and the mapping runs with SQL ELT optimization, the function returns the current date and time based on the time zone associated with the Google BigQuery database.
  When you push the SYSDATE() function to the Google BigQuery database and the mapping runs without SQL ELT optimization, the function returns the current date and time based on the time zone associated with the machine where the Secure Agent runs.

- When you push the SUBSTR() function to the Google BigQuery database, you must specify a value of the String data type in the string argument. If you pass a numeric value, the mapping runs without SQL ELT optimization.

- When you push the SUBSTR() function to the Google BigQuery database, the value of the length argument must be an integer greater than 0. If you specify a negative integer value for the length argument, the mapping runs without SQL ELT optimization.

- When you push the EXP() function to the Google BigQuery database and specify a value of the Numeric or Double data type for the exponent argument, you might encounter a data mismatch in the decimal values when you compare a mapping that runs with full SQL ELT optimization and a mapping that runs without SQL ELT optimization.

- When you push the RPAD() or LPAD() function to the Google BigQuery database, you must specify a value of the String data type in the first_string argument. If you specify a value other than the String data type in the first_string argument, the mapping runs without SQL ELT optimization.
- When you push the RPAD() or LPAD() function to the Google BigQuery database and specify an empty string in the second_string or third_string argument, the mapping runs without full SQL ELT optimization.
- When you push the REPLACECHR() function to the Google BigQuery database to write Numeric data to the Google BigQuery target, you can see a data mismatch between the results when you compare a mapping that runs with full SQL ELT optimization against a mapping disabled for SQL ELT optimization.
If the mapping runs with full SQL ELT optimization, the trailing zeroes after decimal is not considered while casting. However, if you run a mapping with disabled SQL ELT optimization, the Secure Agent casts the trailing zeroes after the decimal while casting from NUMERIC to String data types.
- When you push the REPLACECHR() or REPLACESTR() function to the Google BigQuery database, the microseconds available in the timestamp data is considered in the casted string for a mapping that runs with full SQL ELT optimization.
You might encounter a data mismatch when you compare a mapping that runs with full SQL ELT optimization and a mapping that runs without SQL ELT optimization. The microseconds are not considered in the mapping that runs without SQL ELT optimization.
- When you push the REPLACESTR() or REPLACECHR() function to the Google BigQuery database and specify special characters in the format arguments in a nested function, ensure that the nested function does not contain a single backslash. You can use a double backslash in the nested function.
You might encounter a data mismatch when you compare a mapping that runs with full SQL ELT optimization and a mapping that runs without SQL ELT optimization when you use a double backslash in the nested function.
- When you push the REPLACESTR() or REPLACECHR() function using an Expression transformation with the data/time data types to Google BigQuery using full SQL ELT optimization, the default date format of the data/time data types returned for a mapping with SQL ELT optimization is YYYY-MM-DD HH24:MI:SS.US, whereas for a mapping without SQL ELT optimization is MM/DD/YYYY HH24:MI:SS.US. To fix the issue in a mapping without SQL ELT optimization, use the TO_CHAR function to return the string date in the MM/DD/YYYY HH24:MI:SS.US format. For example, to get a similar result, use replacechr(1, TO_CHAR(col6_date), '09','1').
- When you push down the Is_Number() function for Float data types with NaN, -inf, and +inf values, the Is_Number() function returns true.
- When you push the Is_Date() function using an Expression transformation with the YYYY-MM-DD format and data contains data types with the YYYY-MM-DD and YYYY/MM/DD formats , the Is_Date() function returns true only for YYYYY-MM-DD. When you push the Is_Date() function with the YYYY/MM/DD format and data contains data types with the YYYY-MM-DD and YYYY/MM/DD formats , the Is_Date() function returns true only for YYYY/MM/DD.
- When you push the Is_Number() function to process in Google BigQuery from a Secure Agent machine on Windows, the Is_Number() function returns false for the following format: '45.45d-2'
- When you use the TO_CHAR(String) function, the string value must not contain a backslash (\). Else, the mapping fails.
- When you configure an IN function that returns a value of string data type in an Expression transformation and writes the value to an integer data type in full SQL ELT optimization, the mapping fails with the following error:
  ```
  The Secure Agent failed to run the full SQL ELT query due to the following error: [Bad
  int64 value: false]
  ```
- When you configure an IN function that returns a value of integer data type in an Expression transformation and writes the value to an integer data type in full SQL ELT optimization, the mapping fails with the following error:

```
The Secure Agent failed to run the full SQL ELT query due to the following error: [Query
column 1 has type BOOL which cannot be inserted into column COL_INT, which has type INT64
at [3:4]]
```

- When you configure an IN function that returns a value of integer data type in an Expression
transformation and writes the value to any data type in the target in source SQL ELT optimization, the
mapping fails with the following error:
```
The following error occurred: [For input string: "true"]
```

- When you run a mapping enabled with full SQL ELT optimization and if the arguments are null in the
DATE_COMPARE function, the mapping runs without SQL ELT optimization.

- When you configure a mapping enabled with full SQL ELT optimization, the mapping switches to source
SQL ELT optimization or runs without SQL ELT optimization if the operands of +, −, *, or / operators
contain NULL in the expression, aggregator, or filter transformations.

- When you push the REG_REPLACE() function to the Google BigQuery database and specify a backslash (\)
in the column name or a nested function, the mapping fails with the following error:
```
[ERROR] The Secure Agent failed to run the full SQL ELT query due to the following error:
[Cannot parse regular expression: invalid escape sequence: \o]
```

- When you push the REG_REPLACE() function to the Google BigQuery database, ensure that the column
name or nested function does not contain a single backslash. The function only supports single
backslashes that are followed by a digit or another backslash.
You might encounter a data mismatch when you compare a mapping that runs with full SQL ELT
optimization and a mapping that runs without SQL ELT optimization when you use a double backslash in
the nested function.

- You cannot use the following escape sequences with the regular expression in the REG_REPLACE()
function:

Escape sequence	Description
\1	Back reference
\b	Backspace (use \010)
\cK	Control char ^K (For example, use \001)
\e	Escape (use \033)
\g1	Back reference
\g{1}	Back reference
\g{+1}	Back reference
\g{-1}	Back reference
\g{name}	Named back reference
\g<name>	Subroutine call
\g'name'	Subroutine call
\k<name>	Named back reference

Escape sequence	Description
\k'name'	Named back reference
\lX	Lowercase X
\ux	Uppercase x
\L...\E	Lowercase text ...
\K	Reset beginning of $0
\N{name}	Named Unicode character
\R	Line break
\U...\E	Upper case text ...
\X	Extended Unicode sequence
\%d123	Decimal character 123
\%xFF	Hex character FF
\%o123	Octal character 123
\%u1234	Unicode character 0x1234
\%U12345678	Unicode character 0x12345678

- You cannot use the following empty strings with the regular expression in the REG_REPLACE() function:

Empty string	Description
\g	At beginning of subtext being searched
\G	At end of last match
\Z	At end of text, or before newline at end of text
(?=re)	Before text matching re
(?!re)	Before text not matching re
(?<=re)	After text matching re
(?<!re)	After text not matching re
re&	Before text matching re
re@=	Before text matching re
re@!	Before text not matching re

Empty string	Description
`re@<=`	After text matching `re`
`re@<!`	After text not matching `re`
`\zs`	Sets start of match (= `\K`)
`\ze`	Sets end of match
`\%^`	Beginning of file
`\%$`	End of file
`\%V`	On screen
`\%#`	Cursor position
`\%'m`	Mark `m` position
`\%23l`	In line 23
`\%23c`	In column 23
`\%23v`	In virtual column 23

- When you configure the DATE_DIFF function in an Expression transformation and the transformation output is set to double or decimal data type in a mapping enabled with SQL ELT optimization, the function returns an integer value. However, when you run the mapping without SQL ELT optimization, the function returns a double value. Also, you might encounter a data mismatch when the transformation output is set to integer data type.
- When you set the optional property `OptimizeCastsInPDO` in the Google BigQuery connection, you can compare the following data types in the DATE_DIFF function:

Date1 argument	Date2 argument	Format argument
Date	Date	Year, Month, and Day
Date	Datetime	Year, Month, Day, Hour, Minute, Second, Millisecond, and Microsecond
Datetime	Date	Year, Month, Day, Hour, Minute, Second, Millisecond, and Microsecond
Datetime	Datetime	Year, Month, Day, Hour, Minute, Second, Millisecond, and Microsecond
Time	Time	Hour, Minute, Second, Millisecond, and Microsecond
Timestamp	Timestamp	Day, Hour, Minute, Second, Millisecond, and Microsecond

- When you set the format argument to year, month, nanosecond, or null in the DATE_DIFF function, the mapping runs with source SQL ELT optimization or without SQL ELT optimization.
- When you configure the MD5 function in a mapping enabled with SQL ELT optimization, use BASE64 semantics as the string format. However, when you run the mapping without SQL ELT optimization, use BASE16 semantics as the string format.

- When you configure DECODE or IFF functions along with AND, OR, or NOT IN operators in a mapping enabled with SQL ELT optimization, the mapping might fail.

Supported operators

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent operators in the database. If there is no equivalent operator, the Secure Agent processes the transformation logic.

The following table lists the SQL ELT optimization operators that you can push to the Google BigQuery database by using full SQL ELT optimization:

Operator	Operator
+	=
-	>=
*	<=
/	!=
%	AND
||	OR
>	NOT
<	

Rules and guidelines

When the argument in the NOT operator is a null value, the mapping runs with source SQL ELT optimization or without SQL ELT optimization.

Supported transformations

When you configure SQL ELT optimization, the Secure Agent tries to push the configured transformation to Google BigQuery.

The following list summarizes the availability of transformations that you can push down to Google BigQuery:

- Aggregator
- Expression
- Filter
- Joiner
- Lookup
- Rank
- Router
- Sorter
- SQL

- Union

Aggregator transformation

You can configure full SQL ELT optimization to push an Aggregator transformation to process in Google BigQuery.

You can perform the following aggregate calculations:

- AVG
- COUNT
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

When you configure an Aggregator transformation, you must use each of the incoming ports either in an aggregate function or in a group by field to define how to group data for aggregate expressions.

Lookup transformation

You can configure full SQL ELT optimization to push a Lookup transformation to process in Google BigQuery. This applies to both connected and unconnected lookups.

You can add the following lookups:

- Cached
- Uncached
- Unconnected with cached

When you look up data and the lookup condition finds multiple matches, the lookup returns all rows. In a mapping with Google BigQuery as target, you must set the **Multiple Matches** option for the lookup object to **Return all rows**. If you enabled **Multiple Matches** to any option other than **Return all rows**, the agent ignores it.

When you configure a Lookup transformation based on a Google BigQuery source, adhere to the following guidelines:

- If there are null values in a lookup column, the mapping does not push the rows with null values to the Google BigQuery target. However, if you run the mapping without full SQL ELT optimization, the rows with null values are written to the target.
- When you specify multiple lookup conditions, ensure at least one of the lookup condition uses the Equals operator.
- Ensure that you specify the same Google BigQuery region ID for the source, lookup, and target connection.
- When you use a Lookup transformation, ensure that you select the **Lookup caching enabled** property in the lookup advanced properties.
- When you use an unconnected lookup and use an Expression transformation to assign the unconnected Lookup transformation output to a variable port, the mapping runs without SQL ELT optimization.
- When you use a completely parameterized lookup condition where the input parameter holds the default value and you specify an override from the task using the parameter file, the task does not honor the override and runs with the default value.

When you configure an unconnected Lookup transformation, consider the following rules:

- You must select the **Multiple Matches** property value as **Report error** in the unconnected lookup properties for SQL ELT optimization to work.
- You can only configure an Expression transformation for an output received from an unconnected lookup.

SQL transformation

You can use an SQL transformation to push supported scalar functions to Google BigQuery. When you configure SQL ELT optimization for a mapping, you can use Java or SQL user-defined functions (UDFs) in a SQL transformation and run queries with the Google BigQuery target endpoint.

You can use only the SELECT clause SQL statement to push down a function. The following snippet demonstrates the syntax of a simple SELECT SQL query:

```
SELECT <function_name1>(~Arg~), <function_name2> (~Arg~)...
```

You can push a SQL transformation with the following restrictions:

- You can configure only a SQL query in the SQL transformation. You cannot enable a stored procedure when you push down to Google BigQuery.
- The SQL query must be a simple SELECT statement without 'FROM' and 'WHERE' arguments. The SQL transformation only supports functions with simple SELECT statement.
- You can only use a SQL transformation when the SELECT statement is present in the query property. Even if an entire query containing the SELECT statement comes from a parameterized input port, the SQL ELT optimization fails.
- If any SQL error occurs, the error is added to the `SQLError` field by default. However, when you run a mapping enabled with SQL ELT optimization, the `SQLError` field remains as Null.
- The `NumRowsAffected` field records the number of rows affected while computing the output buffer. However, for SQL transformation, the `NumRowsAffected` is 0, as the query runs for all the records at the same time.
- Google BigQuery offers only passive behavior of SQL transformations where the support for dynamic queries are limited.
- User defined functions containing special characters in its function name are supported. You need to enclose the Full UDF function name with backtick (`) character if it contains special characters.
- You cannot specify the user defined functions in a legacy SQL query.
- You cannot use sub-query and join condition in the SQL transformation.
- You cannot use temporary UDF in the SQL transformation.
- You cannot use the following parameterization scenarios:
  - Entire query as a parameter
  - Field names in a query as a parameter
  - In-out and input parameters in a query

Supported variables

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent variables in the database. If there is no equivalent variable, the Secure Agent processes the transformation logic.

The following table lists the SQL ELT optimization variables that can be used in an Google BigQuery database. Columns marked with an X indicate that the variable can be pushed to the Google BigQuery database by using full SQL ELT optimization.

Variable	SQL ELT optimization
SESSSTARTTIME	X
SYSDATE	-
WORKFLOWSTARTTIME	-

Supported data types

The following table lists the Google Cloud Storage data types based on the file format type that can be pushed to the Google BigQuery database:

File format type	Google Cloud Storage data type
Delimited	BigInt, Number, String
Avro	Binary, Byte, Double, Float, Int, Long, String
Parquet	Binary, Date, Decimal, Double, Float, Int32, Int64, Int96, String
JSON	Double, Int, Long, String

The following table lists the Google BigQuery native data types that can be mapped to the comparable transformation data types in a mapping configured with SQL ELT optimization:

Google BigQuery data type	Transformation data type
Boolean	String
Date	Date/Time
DateTime	Date/Time
Float	Double
Integer	BigInt
Numeric	Decimal Default precision 28, scale 9.
String	String

Google BigQuery data type	Transformation data type
Byte	Byte
Time	Date/Time
Timestamp	Date/Time

When you set the custom property `OptimizeCastsInPDO:true` in Google BigQuery V2 connection optional properties, you can map the following date time data types in Google BigQuery source to a target in a mapping enabled with SQL ELT optimization:

Source data type	Supported target data type
Date	Date, Date/Time
Time	Time
Date/Time	Date, Time, Date/Time
Timestamp	Time, Timestamp

Read from and write to Google BigQuery

You can configure SQL ELT optimization in a mapping to read from and write to Google BigQuery using a Google BigQuery V2 connection.

Example

You work in a motorbike retail company with more than 30,000 dealerships and 2000 inspection centers globally. The company stores millions of records in Google BigQuery hosted on GCP. You want to use Data Integration to perform some transformations on the data before you write back to Google BigQuery.

Use a Google BigQuery V2 connection in the mapping to read from the Google BigQuery source and write the processed data to the Google BigQuery target. Configure full SQL ELT optimization in the mapping to enhance the performance.

Supported features

You must configure a Google BigQuery V2 connection with simple or hybrid mode when you enable SQL ELT optimization in a mapping task.

Note: If you configure a Google BigQuery V2 connection with complex mode, the Secure Agent logs an SQL ELT optimization validation error in the session logs file and the mappings run in the Informatica runtime environment without full SQL ELT optimization.

When you configure SQL ELT optimization, the mappings support the following advance properties for a Google BigQuery V2 source:

- Source type - Single, Query, Multiple Objects, and Parameter

- Query options - Filter. Supports both simple and advanced filter conditions. You can use both the source filter in conjunction with the Filter transformation in the mapping.
- Source Dataset ID
- Source Table Name
- Number of Rows to Read
- Job Poll Interval In Seconds
- Pre SQL using standard SQL query
- Pre SQL Configuration
- Post SQL using standard SQL query
- Post SQL Configuration
- SQL Override Query using standard SQL query
- Billing Project ID

When you configure SQL ELT optimization, the mappings support the following advance properties for a Google BigQuery V2 connected and unconnected lookup:

- Source type - Single
- Source type - Query
- Source type - Standard and materialized views
- Source Object Type - Parameter
- Source Dataset ID
- Source Table Name
- Job Poll Interval In Seconds
- Pre SQL using standard SQL query
- Pre SQL Configuration
- Post SQL using standard SQL query
- Post SQL Configuration
- SQL Override Query using standard SQL query
- Billing Project ID

When you configure SQL ELT optimization, the mappings support the following properties for an Google BigQuery V2 target:

- Target Object Type - Single, Parameter
- Operation
  - Insert
  - Update
  - Upsert
  - Delete
  - Data Driven

  Note: You can implement the Update Strategy through target operations.
- Data Driven Condition
- Target Dataset ID

- Target Table Name

- Update Mode

- Enable Merge

- Create Disposition for Insert operation. Supports only **Create if never** option.

- Write Disposition for Insert operation. Supports only **Write append** option.

- Write Mode. Use Bulk mode to push data into Google BigQuery.

- Truncate target table

- Job Poll Interval In Seconds

- Pre SQL using standard SQL query

- Pre SQL Configuration

- Post SQL using standard SQL query

- Post SQL Configuration

- Billing Project ID

Note: If you configure target advanced properties that are not supported, the Secure Agent logs an validation error in the session logs and the mappings run in the Informatica runtime environment without full SQL ELT optimization.

Rules and guidelines for mappings that read from and write to Google BigQuery

When you configure SQL ELT optimization in a mapping that reads from and writes to Google BigQuery, consider the following guidelines:

- When you perform an upsert operation, you must select the Enable Merge option in the target advanced properties.

- You cannot use system variables in filters.

- If a mapping contains a Filter transformation and also a filter in the Source transformation, the mapping consolidates the filter conditions from both these transformations to filter the records. However, it is recommended that you use only one of these filters at a time in a mapping.

- You cannot apply a filter for query and multiple source objects.

- A native filter cannot contain a sub-query.

- When you select the source type as **Query**, ensure that you do not select the **Retain existing fields at runtime** option on the **Fields** tab. Otherwise, the mapping fails with the following error:
  ```
  Error: The Secure Agent failed to run the full SQL ELT query due to the following error:
  [Field not found inside table]
  ```

- When you configure a Target transformation in a mapping with a delete operation and the source type uses a query that contains Union ALL, the mapping fails. To avoid this error, before you run the mapping, you need to select the **Enable Merge** property in the target advanced properties. The mapping issues a merge query and runs successfully.

- If the source data that the mapping read contains the Binary data types, data preview for SQL ELT optimization fails.

Read from Google Cloud Storage and write to Google BigQuery

You can configure SQL ELT optimization for a mapping that uses a Google Cloud Storage connection in the Source transformation to read from Google Cloud Storage and a Google BigQuery V2 connection in the Target transformation to write to Google BigQuery.

Example

You work for a rapidly growing data science organization. Your organization develops software products to analyze financials, building financial graphs connecting people profiles, companies, jobs, advertisers, and publishers. The organization uses infrastructure based on Google Cloud Platform and stores its data in Google Cloud Storage files. The organization plans to implement a business intelligence service to build visualization and perform real-time analysis. You can load data from Google Cloud Storage to Google BigQuery by configuring the transformations to support the adequate data warehouse model and the consuming requirements.

Create an Google Cloud Storage V2 connection to read data form the Google Cloud Storage source. Create an Google BigQuery V2 connection and use SQL ELT optimization to write data to the Google BigQuery target to enhance the performance and reduce the cost involved.

Supported features

When you configure SQL ELT optimization, the Google Cloud Storage V2 connection supports the following properties:

- Service Account ID
- Service Account Key
- Project ID

When you configure SQL ELT optimization, the mappings support the following properties for a Google Cloud Storage V2 source:

- Source connection, connection parameter
- Source Type - Single, parameter
- Parameter
- Format - Delimited, Avro, Parquet, and JSON. ORC is not applicable.

  Note: None is not supported.
- Delimited file formatting options
  - Delimiter
  - Qualifier
  - Header Line Number
  - First Data Row
- Google Cloud Storage Path
- Source File Name
- Is Directory

Rules and guidelines for mappings that read from Google Cloud Storage V2 source

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping that reads from a Google Cloud Storage V2 source and writes to a Google BigQuery target:

- The source fields must start with a letter or an underscore and can contain letters, numbers, and underscores up to a maximum of 300 characters. You cannot read source fields with special characters.

- When you read a boolean integer column and write to a boolean string column in a mapping, the mapping fails.

- When you write data with the Numeric data types to a Google BigQuery target created at runtime, where the source column has precision greater than 28, the mapping runs without SQL ELT optimization.

- When you set the JVM option system property for the DTM type to `-DHonorInfaDateFormat=true` for the Secure Agent and configure a mapping with SQL ELT optimization, the mapping fails with the following error if it reads the date data type that is not in the YYYY-MM-DD format:
  ```
  The Secure Agent failed to run the full SQL ELT query due to the following error: [Failed
  to parse input string "1972-12-31"]
  ```

- When you map a string data type in the source to a time data type in the target, and the data is in the format HH24:MI:SS.US, the mapping fails with the following error:
  ```
  [Invalid timestamp: '00:00:00.000001']
  ```

Read from Amazon S3 and write to Google BigQuery

You can configure SQL ELT optimization for a mapping that uses an Amazon S3 V2 connection in the Source transformation to read from Amazon S3 and a Google BigQuery connection in the Target transformation to write to Google BigQuery.

Example

You work for a healthcare organization. Your organization offers a suite of services to manage electronic medical records, patient engagement, telephonic health services, and care coordination services. The organization uses infrastructure based on Amazon Web Services and stores its data on Amazon S3. The management plans to load data to a data warehouse to perform healthcare analytics and create data points to improve operational efficiency. To load data from an Amazon S3 based storage object to Google BigQuery, you must use ETL and ELT with the required transformations that support the data warehouse model.

Use an Amazon S3 V2 connection to read data from a file object in an Amazon S3 source and a Google BigQuery connection to write to a Google BigQuery target. Configure full SQL ELT optimization in the mapping to optimize the performance.

Amazon S3 prerequisites

You need to complete the following prerequisites in Amazon S3 before you can read data from an Amazon S3 source and write to Google BigQuery:

- To load data from an Amazon S3 data source, you must:
  - Specify the URI for the Amazon S3 source.
  - Provide your access key ID and secret access key to access the Amazon S3 bucket.

- Set the minimum required policy AmazonS3ReadOnlyAccess on your Amazon S3 source data.

- Enable the BigQuery Data Transfer Service for your project. To enable the BigQuery Data Transfer Service, you must be granted the owner role for your project.

For more information on configuring these prerequisites in Amazon S3, see the Amazon S3 documentation.

Supported features

When you configure SQL ELT optimization, the following connection properties of Amazon S3 V2 source are supported:

- Access Key
- Secret Key
- Folder Path

When you configure SQL ELT optimization, the mappings support the following properties for an Amazon S3 V2 source:

- Source connection parameter
- Source Type - Single, Parameter
- Object
- Parameter
- Format - Flat, Avro, Parquet, and JSON. ORC and None are not applicable.
- Flat file formatting options:
   - Delimiter
   - First Data Row
- Source Type
- Folder Path
- File Name

When you configure SQL ELT optimization, the mapping supports the following transformations:

- Expression
- Filter

Note: You can run a mapping that reads from an Amazon S3 source and writes to a Google BigQuery target, both belonging to different regions.

For information about the configurations for the listed options, see the help for the Amazon S3 V2 Connector.

Rules and guidelines for mappings that read from Amazon S3 source

Use the following rules and guidelines when you configure SQL ELT optimization in a mapping that reads from an Amazon S3 source and writes to a Google BigQuery target:

- Do not map the boolean data type in Amazon S3 to the boolean data type in Google BigQuery. Else, the mapping fails.
- When you edit the metadata in the mapping, you cannot add or remove source fields or change the scale and precision of data types. However, you can edit the field data types.

- When you read data in AVRO, JSON, or CSV format, ensure that the date is in YYYY-MM-DD format and time is in hh:mm:ss format in the DATE, TIME, DATETIME, and TIMESTAMP columns.

- The source fields must start with a letter or an underscore and can contain letters, numbers, and underscores up to a maximum of 300 characters. You cannot read source fields with special characters.

- When you write data with the Numeric data types to a Google BigQuery target created at runtime, where the source column has precision greater than 28, the mapping runs without SQL ELT optimization.

- When you write the DATE, TIME, or DATETIME data types to a Google BigQuery target, you must match the agent time zone with the time zone of the Google BigQuery application.

- In a mapping enabled with SQL ELT optimization, you cannot read a single directory from multiple subdirectories. When you select the source type as **Directory** in the advanced source properties to read objects stored in subdirectories from an Amazon S3 source, you must select the **Enable Recursive Read** option. Otherwise, the mapping runs without SQL ELT optimization.

- When you write data from Avro or Parquet file formats in an Amazon S3 source to a Google BigQuery target created at run time, you must delete the Filename field in the mapping.

- When you configure a lookup from an Amazon S3 or a Google Cloud Storage V2 object in a mapping, the mapping runs without SQL ELT optimization.

- When you read data from a smaller dataset such as Transaction Processing Council Ad-hoc/decision support benchmark (TPC-H) scale factor 1 or below and run a mapping enabled with SQL ELT optimization, the mapping takes 30% more time to process the data as compared to the mapping that runs without SQL ELT optimization.

- When you read from an Amazon S3 source and write to a Google BigQuery target, the time taken to load data to the Google BigQuery staging in the first and subsequent mapping runs for the same dataset and resources is inconsistent.

- When you configure a mapping enabled with SQL ELT optimization to read a boolean integer column and write to a boolean string column, the mapping fails.

- When you read data from an Amazon S3 source and write to a Google BigQuery target, it takes a few minutes to initialize the transfer to the Google BigQuery target.

- When you upload a file in an Amazon S3 bucket and then immediately run a data transfer task, the source file is not detected. Wait for at least five minutes and then run the mapping again.

- When you run a mapping enabled with SQL ELT optimization to read data with wildcard characters from an Amazon S3 source and write to Google BigQuery, the mapping runs without SQL ELT optimization and fails with the following error:
  ```
  Wild card character option is not valid in the native mode of execution
  ```

- When you set the JVM option system property for the DTM type to `-DHonorInfaDateFormat=true` for the Secure Agent and configure a mapping with SQL ELT optimization, the mapping fails with the following error if it reads the date data type that is not in the YYYY-MM-DD format:
  ```
  The Secure Agent failed to run the full SQL ELT query due to the following error: [Failed to parse input string "1972-12-31"]
  ```

- When you map a string data type in the source to a time data type in the target, and the data is in the format HH24:MI:SS.US, the mapping fails with the following error:
  ```
  [Invalid timestamp: '00:00:00.000001']
  ```

Rules and guidelines for SQL ELT optimization

Certain rules and guidelines apply when you enable a mapping for SQL ELT optimization to a Google BigQuery database.

When you configure a Google BigQuery source, Google Cloud Storage source, or Google BigQuery target, adhere to the following guidelines:

- The Service Account ID associated with the Google BigQuery V2 connection must have permissions to access Google Cloud Storage buckets and files.
- You cannot enable full SQL ELT optimization for a mapping task when the target table contains columns of record data type or repeated columns.
- You cannot enable full SQL ELT optimization for a mapping task when the task contains a mapping with a single transformation connected to multiple transformations downstream or multiple transformations connected to a single transformation.
- You must ensure that the column header names in the Google Cloud Storage source file does not contain unicode characters. Otherwise, the mapping fails.
- When you enable SQL ELT optimization for a mapping with multiple pipelines to write to the same Google BigQuery target and the Truncate target table option is enabled in each pipeline, the target table is truncated for each pipeline when data is inserted into the target.
  For example, if there are two pipelines, in pipeline 1 the target table is truncated and then the data is inserted. Similarly, in Pipeline 2 the target table is truncated and the data is inserted into the target table. Hence, the target table contains only the data from pipeline 2.

  When you run a mapping without SQL ELT optimization and the mapping contains multiple pipelines, the target tables are truncated at once for all pipelines and then the data is inserted.
- When you configure a Filter transformation or specify a filter condition, you must ensure that you do not specify special characters. Use the ASCII value for the special character in the filter condition.
- When you parameterize the Google BigQuery V2 connection, source, target and provide values in the mapping task using a parameter file, the default values for the parameter are not overridden with the values in the parameter file.
- If the Google Cloud Storage source file contains a column of Boolean data type, SQL ELT optimization query fails.
- You must ensure that the Google BigQuery source object does not contain any partitions.
- When you read from a Google BigQuery source and edit the metadata for the source fields, the Secure Agent ignores the changes to the metadata.
- If the Google BigQuery source and target object resides in the same region other than US, do not specify the **Region ID** explicitly in the connection.
- You must not specify a legacy SQL query in the **Pre SQL** and **Post SQL** advanced source or target properties.
- A mapping run without SQL ELT optimization fails if any of the Pre-SQL and Post-SQL commands fail in a multi-statement query. Previously mappings were successful.
- When you specify custom query as a source object and specify a dataset name in the **Source Dataset ID** source advanced property, the mapping runs without full SQL ELT optimization.
- When you specify custom query as a source object and specify an SQL override query, you must specify a dataset name in the **Source Dataset ID** source advanced property.
- When you specify custom query as a source object and specify an SQL override query with different column names, ensure that the data types and the order of the columns that appear in the SQL override query matches the data types and order in which they appear in the custom query.

- When you select a view as a source object that contain columns of the Record data type or repeated columns and create a new target at runtime, a validation error appears in the session logs and the mapping runs without full SQL ELT optimization.

- To load data into columns of date, time, datetime, or timestamp in a Google BigQuery target, you must pass the data through the TO_DATE() function as an expression and map the results to the target column.

- When you set SCD Type 2 merge optimization context for a mapping, you cannot use filter, joiner, and custom SQL query.

- If the mapping contains a Router transformation output connected to a Sequence Generator transformation, the mapping does not push down the mapping logic to the point where the transformation is supported and runs without SQL ELT optimization.

- When you push down a Router transformation with IIF and IS_SPACE() functions in mapping that reads from and writes to Google BigQuery, and the Boolean values are 0 and 1, the mapping fails. When the Boolean values are true and false, the mapping runs successfully.

- When you use multiple functions within a transformation and one of the functions cannot be pushed to Google BigQuery, the mapping runs without SQL ELT optimization.

- When the mapping contains multiple pipelines and a function within one of transformations cannot be pushed to Google BigQuery, the mapping does not push down the mapping logic to the point where the transformation is supported and the mapping runs without SQL ELT optimization.

- When you read from or write data to Google BigQuery objects associated with different projects in different Google service accounts that resides in different regions, the mapping runs without SQL ELT optimization.

- When you use the data driven operation to write data to a Google BigQuery target and enable the **Disable Duplicate Update Rows** target advanced property, the Secure Agent ignores the **Disable Duplicate Update Rows** property.

- When you read data from a Google BigQuery source that contains duplicate update keys and enable the **Disable Duplicate Update Rows** target advanced property, the Secure Agent ignores the **Disable Duplicate Update Rows** property.

- When you configure a mapping that includes any of the following datetime scenarios, the mapping runs without SQL ELT optimization:

  - Map data from the TIME data type to any other date/time data type

  - Map data from the DATE data type to the TIME data type

  - Compare data of the TIME and TIMESTAMP data types with the DATE or DATETIME data types

- When you use $$$SESSSTARTTIME variable in a custom query, the variable returns the session start time as a string value. Use the following syntax to convert the string values to timestamp or datetime:

  - SELECT PARSE_DATETIME('%m/%d/%Y %H:%M:%E6S', '$$$SESSSTARTTIME') as t1;

  - SELECT cast(substr(cast('$$$SESSSTARTTIME' as string),0,19) as datetime FORMAT 'MM/DD/ YYYY HH24:MI:SS') as datetime;

  Ensure that the time zones of the Google BigQuery project and the agent machine are the same.

- When you set the OptimizeCastsInPDO:true custom property in the advanced settings for a Google BigQuery V2 connection, the SESSSTARTTIME, SYSDATE, SYSTIMESTAMP, and TO_DATE functions return data of DATETIME data type.

- When you configure a native filter in the Source transformation, ensure that you do not prefix the field name with the table name and dataset name in the filter expression. Otherwise, the mapping fails.

- When you configure an insert operation and set the **Write Disposition** property as Write Truncate in the target transformation properties, the mapping appends the records to the target table instead of truncating the target table before loading data. To configure a truncation when you insert records, you need to select the **Truncate Target Table** option in the target advanced properties.

Troubleshooting a SQL ELT optimization task

Mapping fails when configured to read date, timestamp, or datetime information and write to default date/time format

When you configure a mapping to read date, timestamp, or datetime information from a string column and process the data with the default date/time format to write to Google BigQuery target, the mapping fails with the following error:

```
[ERROR] The Secure Agent failed to run the full SQL ELT query due to the following error:
[Invalid timestamp: '12/31/1972 00:00:00.000001']
```

To resolve this issue, set the JVM option -DHonorInfaDateFormat=true for the Secure Agent.

Perform the following steps to configure the JVM option in Administrator:

1. Select **Administrator** > **Runtime Environments**.
2. On the Runtime Environments page, select the Secure Agent machine that runs the mapping.
3. Click **Edit**.
4. In the System Configuration Details section, select **Data Integration Server** as the Service and **DTM** as the Type.
5. Edit the JVMOption system property and set the value to -DHonorInfaDateFormat=true.
6. Click **Save**.

Mapping fails when configured to read time data in string data type and write to date/time data type

When you set the JVM option system property for the DTM type to -DHonorInfaDateFormat=false for the Secure Agent and also set the optional property OptimizeCastsInPDO=true in the Google BigQuery connection and run a mapping to read the time data in string data type from an Amazon S3 or Google Cloud Storage source and write to date/time data type, the mapping fails with the following error:

```
The Secure Agent failed to run the full SQL ELT query due to the following error:
[Invalid timestamp: '12/31/1972 00:00:00.000001']
```

To resolve this issue, set the JVM option -DHonorInfaDateFormat=true for the Secure Agent and also provide the source data with the format provided in the **DateTime Format String** advanced session property in the mapping task.

Perform the following steps to configure the JVM option in Administrator:

1. Select **Administrator** > **Runtime Environments**.
2. On the Runtime Environments page, select the Secure Agent machine that runs the mapping.
3. Click **Edit**.
4. In the System Configuration Details section, select **Data Integration Server** as the Service and **DTM** as the Type.
5. Edit the JVMOption system property and set the value to -DHonorInfaDateFormat=true.
6. Click **Save**.

CHAPTER 10

Data type reference

Data Integration uses the following data types in mappings and mapping tasks with Google BigQuery:

Google BigQuery native data types

> Google BigQuery data types appear in the **Fields** tab for Source and Target transformations when you choose to edit metadata for the fields.

Transformation data types

> Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the Secure Agent uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

> When Data Integration reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When Data Integration writes to a target, it converts the transformation data types to the comparable native data types.

Google BigQuery V2 and transformation data types

The following table describes the data types that Data Integration supports for Google BigQuery sources and targets:

Google BigQuery Data Type	Transformation Data Type	Range and Description for the Transformation Data Type
BOOLEAN	String	Boolean True or False values. Default precision is 5.
BIGNUMERIC	String or Decimal	The BigNumeric data type by default maps to the String data type. You can use edit metadata option to map the BigNumeric data type to Decimal data type in the Source and Target transformations. For String data type: - 1 to 104,857,600 characters. - Default precision is 255. - You can increase the value up to 104,857,600 characters. For Decimal data type: - Precision 28 and scale 9 for the source. - Precision 29 and scale 9 for the target.

Google BigQuery Data Type	Transformation Data Type	Range and Description for the Transformation Data Type
DATE[1]	Date/Time	Date values. Google BigQuery Connector uses the following format: `YYYY-[M]M-[D]D` Minimum value: 0001-01-01 Maximum value: 9999-12-31 Precision 29, scale 9
DATETIME[1]	Date/Time	Google BigQuery Connector uses the following format: `YYYY-[M]M-[D]D[( |T)[H]H:[M]M:[S]S[.DDDDDD]]` Minimum value: 0001-01-01 00:00:00 Maximum value: 9999-12-31 23:59:59.999999 Default precision 29, scale 9
FLOAT	Double	Precision 15, scale 0
INTEGER	BigInt	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
RECORD	String	1 to 104,857,600 characters Default precision is 255. You can increase the value up to 104857600 characters.

NUMERIC	Decimal	- For mappings:
		  Default precision 28, scale 9.
		  Maximum precision of 29.
		  Range: -9.9999999999999999999999999999999999999E+29 to 9.9999999999999999999999999999999999999E+29
		- For mappings in advanced mode:
		  Default precision 38, scale 9. For hierarchical data types, only up to a precision of 28 digits is applicable.
		  Note: You cannot use the decimal data type for hierarchical data in advanced mode.
		  Range: -9.9999999999999999999999999999999999999E+28 to 9.9999999999999999999999999999999999999E+28

STRING	String	1 to 104,857,600 characters Default precision is 255. You can increase the value up to 104,857,600 characters. **Note:** If you configure the optional connection property **SupportParameterisedDatatype**, you can use the string column precision that is defined in Google BigQuery when you import the metadata. However, if the precision defined in Google BigQuery is higher than 104,857,600 characters, then the precision is updated to 104,857,600 characters.
BYTE	Bytes	1 to 104,857,600 bytes

Google BigQuery Data Type	Transformation Data Type	Range and Description for the Transformation Data Type
TIME[1]	Date/Time	Time values. Google BigQuery Connector uses the following format: `[H]H:[M]M:[S]S[.DDDDDD]` Minimum value: 00:00:00 Maximum value: 23:59:59.999999 Precision 29, scale 9
TIMESTAMP[1]	Date/Time	Google BigQuery Connector uses the following format: `YYYY-[M]M-[D]D[( |T)[H]H:[M]M:[S]S[.DDDDDD]][time zone]` Minimum value: 0001-01-01 00:00:00 Maximum value: 9999-12-31 23:59:59.999999 UTC Precision 29, scale 9
INT64[2]	BigInt	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
FLOAT64[2]	Double	Precision 15, scale 0
BOOL[2]	String	Boolean True or False values. Default precision is 5.

[1]where
- `YYYY` represents four-digit year
- `[M]M` represents one or two digit month
- `[D]D` represents one or two digit day
- `( |T)` represents a space or a `T` separator
- `[H]H` represents one or two digit hour (valid values from 00 to 23)
- `[M]M` represents one or two digit minutes (valid values from 00 to 59)
- `[S]S` represents one or two digit seconds (valid values from 00 to 59)
- `[.DDDDDD]`: represents microseconds up to six fractional digits.
- `[time zone]` represents the time zone. Default time zone is UTC. Other time zones are not applicable.

[2]Applies only to SQL transformation.

Part III: Data Integration with Google BigQuery Connector

This part contains the following chapters:

CHAPTER 11

Introduction to Google BigQuery Connector

You can use Google BigQuery Connector to connect to Google BigQuery from Data Integration. Use Google BigQuery Connector to read data from and write data to Google BigQuery. You can use a Google BigQuery object as a source and as a target in synchronization tasks, mapping tasks, and mappings.

You can switch mappings to advanced mode to include transformations and functions that enable advanced functionality.

When you run a task or mapping, the Secure Agent uses the JAVA client libraries of the Google APIs to integrate with Google BigQuery.

Data Integration Hosted Agent

You can use the Data Integration Hosted Agent (Hosted Agent) as a runtime environment for a Google BigQuery connection if you have the Cloud Runtime license.

Data Integration Secure Agents are installed locally. As an alternative to installing a Secure Agent, you can use a Hosted Agent. Hosted Agents are hosted at Data Integration hosting facility. The Data Integration hosting facility manages the Hosted Agent runtime environment and the agents that run in it. You cannot add, delete, or configure a Hosted Agent runtime environment. Because you do not install a Hosted Agent, you do not have access to files normally stored in the Secure Agent directory, such as configuration, success, and reject files.

Google BigQuery Connector assets

Create assets in Data Integration to integrate data using Google BigQuery Connector.

You can perform insert, update, upsert, and delete operations on a Google BigQuery target.

When you use Google BigQuery Connector, you can include the following Data Integration assets:

- Mapping
- Mapping task
- Synchronization task

For more information about configuring assets and transformations, see *Mappings*, *Transformations*, and *Tasks* in the Data Integration documentation.

Google BigQuery example

Your organization is an open source log data collector, which collects log data from multiple sources and unifies them.

Logs help you understand how systems and applications perform. As the scale and complexity of the system increases, it is difficult to manage multiple logs from different sources.

To overcome this problem, you can use Google BigQuery Connector to write data to a Google BigQuery target and query terabytes of logs in seconds. You can then use the data to fix and improve the system performance in near real time.

Administration of Google BigQuery Connector

Google BigQuery is a RESTful web service that the Google Cloud Platform provides.

Before you use Google BigQuery Connector, you must complete the following prerequisite tasks:

- Create a Google account to access Google BigQuery.

- On the **Credentials** page, navigate to the APIs and auth section, and create a service account. After you create the service account, you can download a JSON file that contains the client_email, project_id, and private_key values. You will need to enter these details when you create a Google BigQuery connection in Data Integration.
  The following image shows the **Credentials** page where you can create the service account and key:

- On the **Dashboards** page of the Google API Console, https://console.developers.google.com/, enable the **BigQuery API** and **Google Cloud Storage JSON API**. Google BigQuery Connector uses the Google APIs to integrate with Google BigQuery and Google Cloud Storage.
  The following image shows the **Dashboard** page where you can enable the APIs:

- Create a project and dataset in Google BigQuery. Verify that the dataset contains the source table and the target table. You will need to enter the project ID, dataset ID, source table name, and target table name when you create tasks and mappings in Data Integration.
  The following image shows a project:
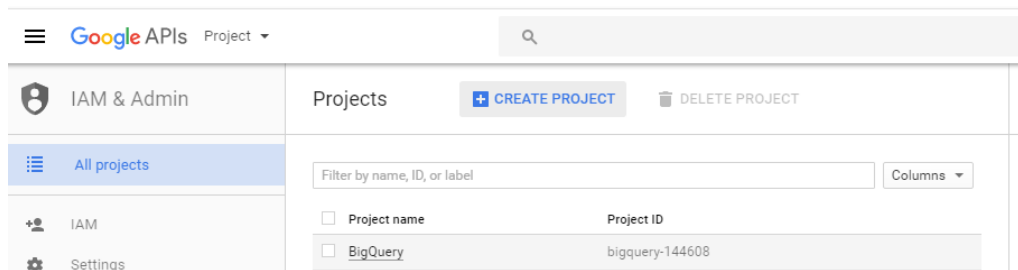
- Verify that you have read and write access to the Google BigQuery dataset that contains the source table and target table.
- When you read data from or write data to a Google BigQuery table, you must have the following permissions:
  - bigquery.datasets.create
  - bigquery.datasets.get
  - bigquery.datasets.getIamPolicy
  - bigquery.datasets.updateTag
  - bigquery.models.*
  - bigquery.routines.*
  - bigquery.tables.create
  - bigquery.tables.delete
  - bigquery.tables.export
  - bigquery.tables.get
  - bigquery.tables.getData
  - bigquery.tables.list
  - bigquery.tables.update
  - bigquery.tables.updateData
  - bigquery.tables.updateTag

- resourcemanager.projects.get

- resourcemanager.projects.list

- bigquery.jobs.create

- When you only read data from a Google BigQuery table, you must have the following permissions:

- bigquery.datasets.get

- bigquery.datasets.getIamPolicy

- bigquery.models.getData

- bigquery.models.getMetadata

- bigquery.models.list

- bigquery.routines.get

- bigquery.routines.list

- bigquery.tables.export

- bigquery.tables.get

- bigquery.tables.getData

- bigquery.tables.list

- resourcemanager.projects.get

- resourcemanager.projects.list

- bigquery.jobs.create

- bigquery.tables.create

- If your organization passes data through a proxy or protective firewall, you must configure your firewall to allow the `www.googleapis.com` URI for Google BigQuery Connector to transfer data through a proxy or firewall.

- If you use bulk mode, verify that you have write access to the Google Cloud Storage path where the Secure Agent creates the staging file.

- If you use staging mode, verify that you have read access to the Google Cloud Storage path where the Secure Agent creates the staging file to store the data from the Google BigQuery source.

CHAPTER 12

Google BigQuery connections

Create a Google BigQuery connection to read data from a Google BigQuery source and write data to a Google BigQuery target. You must create a connection for each dataset that you want to connect to. You can use Google BigQuery connections in synchronization tasks, mapping tasks, and mappings. When you create a Google BigQuery connection, you can configure a connection mode based on how you want to read and write the data.

Connection modes

You can configure a Google BigQuery connection to use one of the following connection modes:
Simple mode

If you use simple mode, Google BigQuery Connector flattens each field within the Record data type field as a separate field in the field mapping.

Hybrid mode

If you use hybrid mode, Google BigQuery Connector displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery Connector displays the top-level Record data type field as a single field of the String data type in the field mapping.

Complex mode

If you use complex mode, Google BigQuery displays all the columns in the Google BigQuery table as a single field of the String data type in the field mapping.

Connection mode example

Google BigQuery Connector reads and writes the Google BigQuery data based on the connection mode that you configure for the Google BigQuery connection.

You have a Customers table in Google BigQuery that contains primitive fields and the **Address** field of the Record data type. The Address field contains two primitive sub-fields, **City** and **State**, of the String data type.

The following image shows the schema of the Customers table in Google BigQuery:

ID	INTEGER	NULLABLE
Name	STRING	NULLABLE
Address	RECORD	NULLABLE
Address.City	STRING	NULLABLE
Address.State	STRING	NULLABLE
Mobile	STRING	REPEATED
Totalpayments	FLOAT	NULLABLE
age	INTEGER	REPEATED

The following table shows the Customers table data in Google BigQuery:

ID	Name	Address.City	Address.State	Mobile	Totalpayments
14	John	LOS ANGELES	CALIFORNIA	+1-9744884744	18433.90
				+1-8267389993	
29	Jane	BOSTON	MANHATTAN	+1-8789390309	28397.33
				+1-9876553784	
				+1-8456437848	

Simple mode

If you use simple connection mode, Google BigQuery Connector flattens each field within the Record data type field as a separate field in the **Field Mapping** tab.

The following table shows two separate fields, Address_City and Address_State, for the respective sub-fields within the Address Record field in the Customers table:

ID	Name	Address_City	Address_State	Mobile	Totalpayments
14	John	LOS ANGELES	CALIFORNIA	+1-9744884744	18433.90
14	John	LOS ANGELES	CALIFORNIA	+1-8267389993	18433.90
29	Jane	BOSTON	MANHATTAN	+1-8789390309	28397.33
29	Jane	BOSTON	MANHATTAN	+1-9876553784	28397.33
29	Jane	BOSTON	MANHATTAN	+1-8456437848	28397.33

The following image shows the fields in the **Field Mapping** tab of a synchronization task:

Hybrid mode

If you use hybrid connection mode, Google BigQuery Connector displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery Connector displays the top-level Record data type field as a single field of the String data type in the **Field Mapping** tab.

The following image shows the **Field Mapping** tab of a synchronization task:

Complex mode

If you use complex connection mode, Google BigQuery Connector displays all the columns in the Google BigQuery table as a single field of the String data type in the **Field Mapping** tab.

The following image shows the STRING_DATA field in the **Field Mapping** tab of a synchronization task:
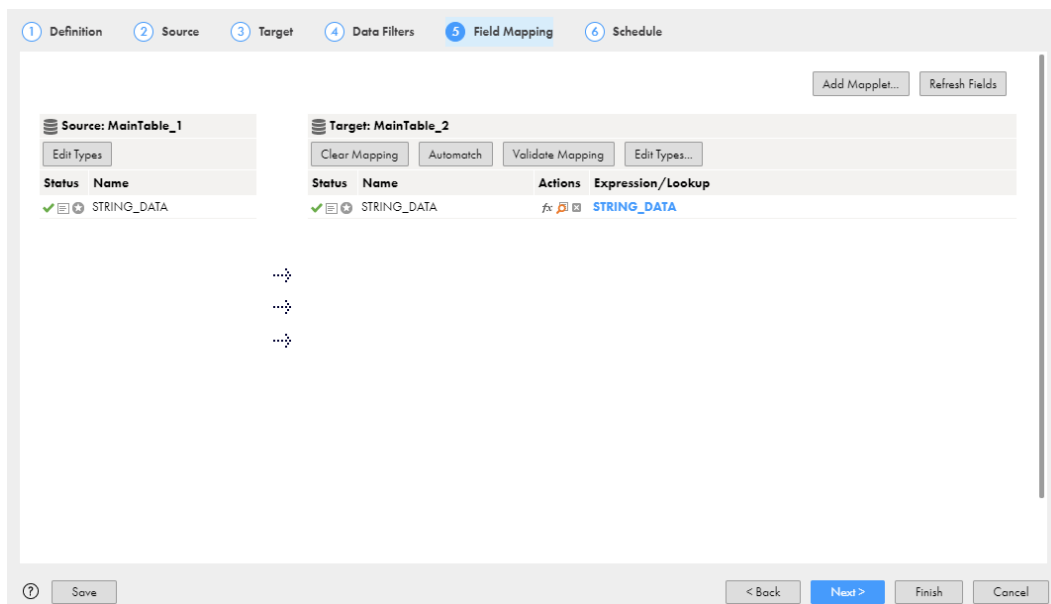
Rules and guidelines for Google BigQuery connection modes

Simple mode

Consider the following rules and guidelines when you configure a Google BigQuery connection to use simple connection mode:

- You cannot create a Google BigQuery target table that contains repeated columns using the **Create Target** option.
- If the Google BigQuery source table contains repeated columns, you cannot configure data filters for these columns.
- If the Google BigQuery table contains more than one repeated column, you cannot preview data.
- If the Google BigQuery target table contains repeated columns, you cannot configure update and delete operations for these columns.
- You cannot configure upsert operations for columns of the Record data type and repeated columns.
- When you read data from a Google BigQuery source, you must not map more than one repeated column in a single mapping. You must create multiple mappings for each repeated column.

Hybrid mode

Consider the following rules and guidelines when you configure a Google BigQuery connection to use hybrid connection mode:

- You cannot preview data.
- You cannot create a Google BigQuery target table using the **Create Target** option.
- If the Google BigQuery source table contains columns of the Record data type and repeated columns, you cannot configure data filters for these columns.
- You cannot configure update, upsert, and delete operations for columns of the Record data type and repeated columns.
- You must select JSON (Newline Delimited) format as the data format of the staging file under the advanced target properties. You can use CSV format as the data format of the staging file unless the Google BigQuery table contains columns of the Record data type or repeated columns.
- The following CSV formatting options in the advanced target properties are not applicable:
  - Allow Quoted Newlines
  - Field Delimiter
  - Allow Jagged Rows

Complex mode

Consider the following rules and guidelines when you configure a Google BigQuery connection to use complex connection mode:

- You cannot preview data.
- You cannot create a Google BigQuery target table using the **Create Target** option.
- When you configure a Google BigQuery source connection to use complex connection mode, you cannot configure data filters for the source.
- You cannot configure update, upsert, and delete operations.
- You must select JSON (Newline Delimited) format as the data format of the staging file under the advanced target properties.

- You cannot use CSV format as the data format of the staging file. The following CSV formatting options in the advanced target properties are not applicable:

  - Allow Quoted Newlines

  - Field Delimiter

  - Allow Jagged Rows

- You cannot use key range partitioning for Google BigQuery sources.

Google BigQuery connection properties

When you create a Google BigQuery connection, you must configure the connection properties.

The following table describes the Google BigQuery connection properties:

Property	Description
Connection Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}]	\ : ; " ' < , > . ? /
Description	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Type	The Google BigQuery connection type.
Runtime Environment	Name of the runtime environment where you want to run the tasks.
Service Account ID	Specifies the client_email value present in the JSON file that you download after you create a service account.
Service Account Key	Specifies the private_key value present in the JSON file that you download after you create a service account.
Connection mode	The mode that you want to use to read data from or write data to Google BigQuery. Select one of the following connection modes: - Simple. Flattens each field within the Record data type field as a separate field in the mapping. - Hybrid. Displays all the top-level fields in the Google BigQuery table including Record data type fields. Google BigQuery Connector displays the top-level Record data type field as a single field of the String data type in the mapping. - Complex. Displays all the columns in the Google BigQuery table as a single field of the String data type in the mapping. Default is Simple.

Property	Description
Schema Definition File Path	Specifies a directory on the Secure Agent machine where the Secure Agent must create a JSON file with the sample schema of the Google BigQuery table. The JSON file name is the same as the Google BigQuery table name. Alternatively, you can specify a storage path in Google Cloud Storage where the Secure Agent must create a JSON file with the sample schema of the Google BigQuery table. You can download the JSON file from the specified storage path in Google Cloud Storage to a local machine. The schema definition file is required if you configure complex connection mode in the following scenarios: - You add a Hierarchy Builder transformation in a mapping to read data from relational sources and write data to a Google BigQuery target. - You add a Hierarchy Parser transformation in a mapping to read data from a Google BigQuery source and write data to relational targets.
Project ID	Specifies the project_id value present in the JSON file that you download after you create a service account. If you have created multiple projects with the same service account, enter the ID of the project that contains the dataset that you want to connect to.
Dataset ID	Name of the dataset that contains the source table and target table that you want to connect to. **Note:** Google BigQuery supports the datasets that reside only in the US region.
Storage Path	This property applies when you read or write large volumes of data. Required if you read data in staging mode or write data in bulk mode. Path in Google Cloud Storage where the Secure Agent creates a local stage file to store the data temporarily. You can either enter the bucket name or the bucket name and folder name. For example, enter `gs://<bucket_name>` or `gs://<bucket_name>/<folder_name>`

Note: Ensure that you specify valid credentials in the connection properties. The test connection is successful even if you specify incorrect credentials in the connection properties.

Configuring the proxy settings on Windows

To configure the proxy server settings for the Secure Agent on a Windows machine, you must configure the proxy server settings through the Secure Agent Manager and the JVM options of the Secure Agent.

Contact your network administrator for the correct proxy settings.

1. Click **Start** > **All Programs** > **Informatica Cloud Secure Agent** > **Informatica Cloud Secure Agent** to launch the Secure Agent Manager.

   The Secure Agent Manager displays the Secure Agent status.

2. Click **Proxy** in the Secure Agent Manager page.

3. Click **Use a Proxy Server** to enter proxy server settings.

4. Configure the following proxy server details:

Field	Description
Proxy Host	Host name of the outgoing proxy server that the Secure Agent uses.
Proxy Port	Port number of the outgoing proxy server.
User Name	User name to connect to the outgoing proxy server.
Password	Password to connect to the outgoing proxy server.

5. Click **OK**.

6. Log in to Informatica Intelligent Cloud Services.

7. Open Administrator and select **Runtime Environments**.

8. Select the Secure Agent for which you want to configure a proxy server.

9. On the upper-right corner of the page, click **Edit**.

10. In the **System Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service.

11. To use a proxy server, add the following parameters in any **JVMOption** field and specify appropriate values for each parameter:

Parameter	Description
-Dproxy.host=	Host name of the outgoing HTTPS proxy server.
-Dproxy.port=	Port number of the outgoing HTTPS proxy server.
-Dproxy.user=	User name for the HTTPS proxy server.
-Dproxy.password=	Password for the HTTPS proxy server.

Note: You must specify the parameter and the value for the parameter enclosed in single quotation marks.

For example,

```
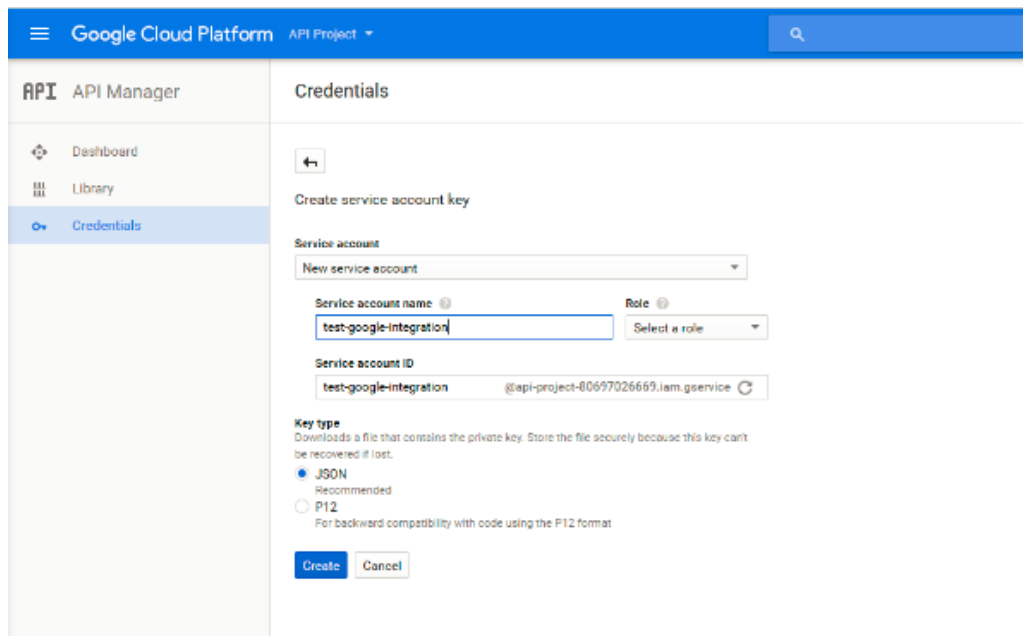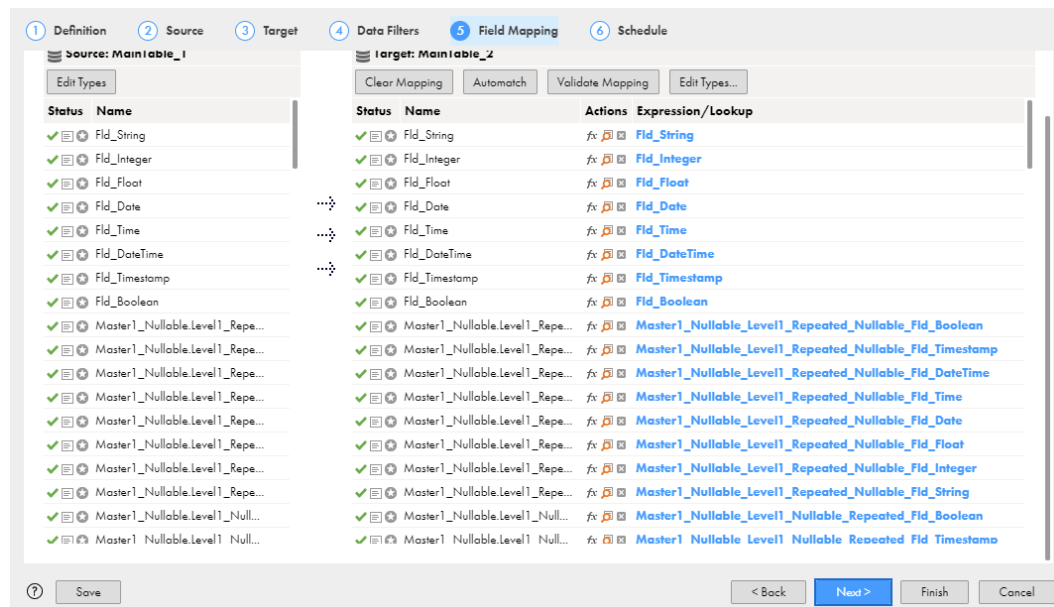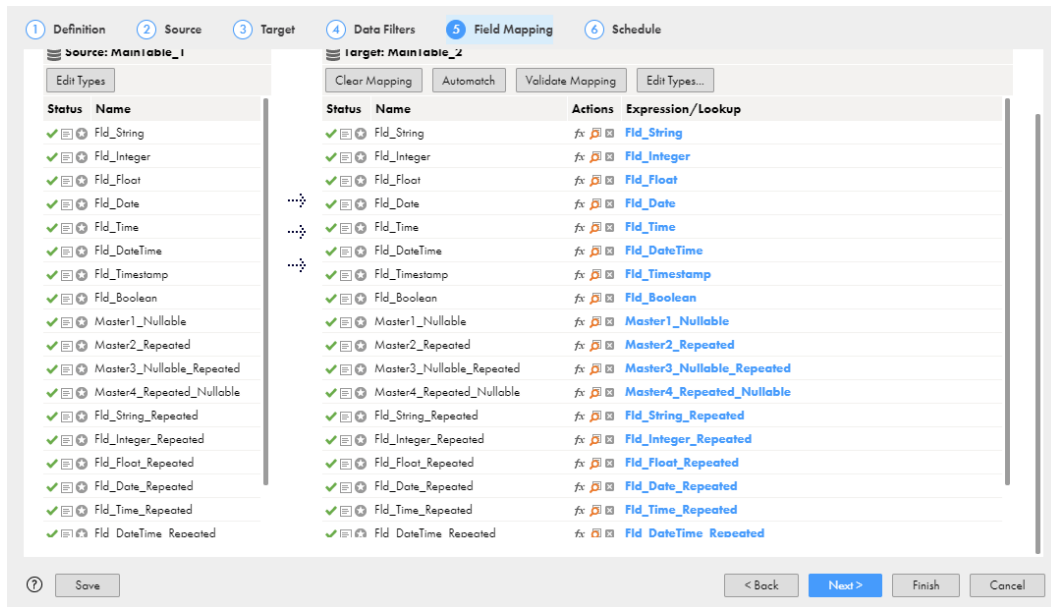JVMOption1='-Dproxy.host=INPQ8583WI29'
JVMOption2='-Dproxy.port=8081'
JVMOption3='-Dproxy.user=adminuser'
JVMOption4='-Dproxy.password=password'
```

Note: You can configure only five **JVMOption** fields in the **System Configuration Details** section. To configure the remaining parameters, you must add the **JVMOption** fields in the **Custom Configuration Details** section. In the **Custom Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service, add the **JVMOption** fields, and specify the remaining parameters and appropriate values for each parameter.

12. Click **Save**.

The Secure Agent restarts to apply the settings.

Note: The session log does not log the proxy server details even if you have configured a proxy server.

Configuring the proxy settings on Linux

You can update the proxy server settings defined for the Secure Agent from the command line. To configure the proxy server settings for the Secure Agent on a Linux machine, you must update the `proxy.ini` file and configure the JVM options of the Secure Agent.

Contact your network administrator for the correct proxy settings.

1. Navigate to the following directory:

   ```
   <Secure Agent installation directory>/apps/agentcore/conf
   ```

2. To update the `proxy.ini` file, add the following parameters and specify appropriate values for each parameter:

   ```
   InfaAgent.ProxyHost=<proxy_server_hostname>
           InfaAgent.ProxyPort=<proxy_server_port>
           InfaAgent.ProxyUser=<user_name>
           InfaAgent.ProxyPassword=<password>
           InfaAgent.ProxyPasswordEncrypted=false
   ```

   For example,

   ```
   InfaAgent.ProxyHost=INW2PF0MT01V
           InfaAgent.ProxyPort=808
           InfaAgent.ProxyUser=user06
           InfaAgent.ProxyPassword=user06
           InfaAgent.ProxyPasswordEncrypted=false
   ```

3. Log in to Informatica Intelligent Cloud Services.

4. Open Administrator and select **Runtime Environments**.

5. Select the Secure Agent for which you want to configure a proxy server.

6. On the upper-right corner of the page, click **Edit**.

7. In the **System Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service.

8. To use a proxy server, add the following parameters in any **JVMOption** field and specify appropriate values for each parameter:

Parameter	Description
-Dproxy.host=	Host name of the outgoing HTTPS proxy server.
-Dproxy.port=	Port number of the outgoing HTTPS proxy server.
-Dproxy.user=	User name for the HTTPS proxy server.
-Dproxy.password=	Password for the HTTPS proxy server.

Note: You must specify the parameter and the value for the parameter enclosed in single quotation marks.

For example,

```
JVMOption1='-Dproxy.host=INPQ8583WI29'
```

```
JVMOption2='-Dproxy.port=8081'
```

```
JVMOption3='-Dproxy.user=adminuser'
```

```
JVMOption4='-Dproxy.password=password'
```

Note: You can configure only five **JVMOption** fields in the **System Configuration Details** section. To configure the remaining parameters, you must add the **JVMOption** fields in the **Custom Configuration Details** section. In the **Custom Configuration Details** section, select the **Type** as **DTM** for the Data Integration Service, add the **JVMOption** fields, and specify the remaining parameters and appropriate values for each parameter.

9. Click **Save**.

The Secure Agent restarts to apply the settings.

Note: The session log does not log the proxy server details even if you have configured a proxy server.

CHAPTER 13

Synchronization Tasks with Google BigQuery Connector

Use the Synchronization task to synchronize data between a source and target.

You can configure a synchronization task using the Synchronization Task wizard.

When you create a task, you can associate it with a schedule to run it at specified times or on regular intervals. Or, you can run it manually. You can monitor tasks that are currently running in the activity monitor and view logs about completed tasks in the activity log.

Pre SQL and post SQL commands

You can specify **pre SQL** and **post SQL** advanced properties for Google BigQuery sources and targets. When you create a task in Data Integration, you can specify SQL commands in the advanced properties for a source and target.

You can perform the following operations by using pre SQL and post SQL commands:

- SELECT
- UPDATE
- DELETE

Note: You cannot perform more than one operation with a pre SQL or post SQL command.

You can configure the options in Google BigQuery with a pre SQL or post SQL statement in the **pre SQL Configuration** or **post SQL Configuration** advanced properties for Google BigQuery sources and targets.

You must use the following format to specify a pre SQL configuration or a post SQL configuration:

```
<Option1:Value1,Option2:Value2,...OptionN:ValueN>
```

The following table shows the configuration options and supported values that you can specify in a pre SQL configuration or post SQL configuration:

Options	Supported Values
DestinationDataset	Dataset ID in Google BigQuery
DestinationTable	Table name in Google BigQuery

Options	Supported Values
FlattenResults	True and False
UseLegacySQL	True and False
WriteDisposition	WRITE_TRUNCATE, WRITE_APPEND, and WRITE_EMPTY

Note: If you perform an UPDATE or DELETE operation with a pre SQL or post SQL command, you must specify the following parameter in the pre SQL configuration or post SQL configuration: `UseLegacySQL:False`

Google BigQuery sources in synchronization tasks

You can use a single object in a synchronization task.

You can configure the Google BigQuery source properties on the **Source** page of the Synchronization Task wizard.

The following table describes the Google BigQuery source properties:

Property	Description
Connection	Name of the active Google BigQuery source connection.
Source Type	Type of the Google BigQuery source objects available. You can read data from a single Google BigQuery source object.
Source Object	Name of the Google BigQuery source object.
Display technical names instead of labels	This property is not applicable for Google BigQuery Connector because both the technical names and labels are the same for Google.
Display source fields in alphabetical order	Displays source fields in alphabetical order. By default, fields appear in the order returned by the source system.

Read modes

When you use Google BigQuery Connector, you can read data by using direct mode or staging mode. Before you choose a mode, see the Google documentation to understand the cost implications and trade-offs for each mode.

You can read data from a Google BigQuery source by using one of the following modes:
Direct mode

Use direct mode when the volume of data that you want to read is small. In direct mode, Google BigQuery Connector directly reads data from a Google BigQuery source. You can configure the number of rows that you want Google BigQuery Connector to read in one request.

Staging mode

Use staging mode when you want to read large volumes of data in a cost-efficient manner.

In staging mode, Google BigQuery Connector first exports the data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery Connector downloads the data from Google Cloud Storage into a local stage file. You can configure the local stage file directory in the advanced source properties. Google BigQuery Connector then reads the data from the local stage file.

When you enable staging file compression, Google BigQuery Connector compresses the size of the staging file in Google Cloud Storage. Google BigQuery Connector then downloads the staging file and decompresses the staging file before it reads the file. To improve the performance and download data in parallel, you can configure the number of threads for downloading the staging file.

Advanced Properties for Google BigQuery sources

You can configure advanced source properties on the **Schedule** page of the Synchronization Task wizard.

The following table describes the advanced properties that you can configure for a Google BigQuery source:

Property	Description
Source Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
Number of Rows to Read	Specifies the number of rows to read from the Google BigQuery source table.
Allow Large Results	Determines whether Google BigQuery Connector must produce arbitrarily large result tables to query large source tables. If you select this option, you must specify a destination table to store the query results.
Query Results Table Name	Required if you select the **Allow Large Results** option. Specifies the destination table name to store the query results. If the table is not present in the dataset, Google BigQuery Connector creates the destination table with the name that you specify.
Job Poll Interval in Seconds	The number of seconds after which Google BigQuery Connector polls the status of the read job operation. Default is 10.
Read Mode	Specifies the read mode to read data from the Google BigQuery source. You can select one the following read modes: - Direct. In direct mode, Google BigQuery Connector reads data directly from the Google BigQuery source table. **Note:** When you use hybrid and complex connection mode, you cannot use direct mode to read data from the Google BigQuery source. - Staging. In staging mode, Google BigQuery Connector exports data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery Connector downloads the data from Google Cloud Storage into the local stage file and then reads data from the local stage file. Default is Direct mode.

Property	Description
Number of Threads for Downloading Staging Files	Specifies the number of files that Google BigQuery Connector downloads at a time to enable parallel download. This property applies to staging mode.
Data Format of the staging file	Specifies the data format of the staging file. You can select one of the following data formats: - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ. **Note:** Avro format is not applicable for Google BigQuery Connector.
Local Stage File Directory	Specifies the directory on your local machine where Google BigQuery Connector stores the Google BigQuery source data temporarily before it reads the data. This property applies to staging mode.
Staging File Name	Name of the staging file where data from the Google BigQuery source table is exported to Google Cloud Storage. This property applies to staging mode.
Enable Staging File Compression	Indicates whether to compress the size of the staging file in Google Cloud Storage before Google BigQuery Connector reads data from the staging file. You can enable staging file compression to reduce cost and transfer time. This property applies to staging mode.
Persist Destination Table	Indicates whether Google BigQuery Connector must persist the query results table after it reads data from the query results table. By default, Google BigQuery Connector deletes the query results table.
pre SQL	SQL statement that you want to run before reading data from the source. For example, if you want to select records in the database before you read the records from the table, specify the following pre SQL statement: `SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;`

pre SQL Configuration	Specify a pre SQL configuration.
	For example,
	`DestinationTable:PRESQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

post SQL	SQL statement that you want to run after reading data from the source.
	For example, if you want to update records in a table after you read the records from a source table, specify the following post SQL statement:
	`UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	`SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where`
	`fullname='John Doe'`

post SQL Configuration	Specify a post SQL configuration.
	For example,
	`DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:True,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

Data filters

You can create simple or advanced data filters. You can also create a set of data filters for each object included in a synchronization task. Each set of data filters act independently of the other sets.

You can create simple or advanced data filters for the following data types:

- Integer
- Float
- Numeric (only if you use a Google BigQuery connection in hybrid mode)
- String
- Timestamp

Simple Data Filters

You can create one or more simple data filters. When you create multiple simple data filters, the associated task creates an AND operator between the filters and loads rows that apply to all simple data filters.

Advanced Data Filters

You can create an advanced data filter to create complex expressions that use AND, OR, or nested conditions. The expression that you enter becomes the WHERE clause in the query used to retrieve records from the source.

Note: Google BigQuery Connector does not support the $LastRunDate variable under advanced data filters.

Google BigQuery targets in synchronization tasks

You can use a single Google BigQuery object as a target in a synchronization task.

The following table describes the Google BigQuery target properties:

Property	Description
Connection	Name of the active Google BigQuery target connection that is associated with a dataset.
Target Object	You can select an existing object from the list or create a target at run time.
Child Object	This property is not applicable for Google BigQuery Connector.
Display technical names instead of labels	This property is not applicable for Google BigQuery Connector because both the technical names and labels are the same for Google.
Display target fields in alphabetical order	Displays target fields in alphabetical order. By default, fields appear in the order returned by the target system.

Write modes

When you use Google BigQuery Connector, you can write data by using bulk mode or streaming mode. Before you choose a mode, see the Google documentation to understand the cost implications and trade-offs for each mode.

You can write data to a Google BigQuery target by using one of the following modes:

Bulk mode

Use bulk mode when you want to write large volumes of data in a cost-efficient manner.

In bulk mode, Google BigQuery Connector first writes the data to a staging file in Google Cloud Storage. When the staging file contains all the data, Google BigQuery Connector loads the data from the staging file to the BigQuery target.

When you enable staging file compression, Google BigQuery Connector compresses the size of the staging file before it writes data to Google Cloud Storage. Google BigQuery Connector writes the compressed file to Google Cloud Storage and then submits a load job to the BigQuery target.

Note: Enabling compression reduces the time that Google BigQuery Connector takes to write data to Google Cloud Storage. However, there will be a performance degradation when Google BigQuery Connector writes data from Google Cloud Storage to the BigQuery target.

Google BigQuery Connector deletes the staging file unless you configure the task or mapping to persist the staging file. You can choose to persist the staging file if you want to archive the data for future reference.

Streaming mode

Use streaming mode when you want the Google BigQuery target data to be immediately available for querying and real-time analysis. Evaluate Google's streaming quota policies and billing policies before you use streaming mode.

In streaming mode, Google BigQuery Connector directly writes data to the BigQuery target. Google BigQuery Connector appends the data into the BigQuery target.

You can configure the number of rows that you want Google BigQuery Connector to stream in one request. If you want to stream a larger number of rows than the maximum permissible limit prescribed by Google, you can write the data to multiple smaller target tables instead of one large target table. You can create a template table based on which Google BigQuery must create multiple tables. You can define a unique suffix for each table. Google BigQuery creates each table based on the template table and adds the suffix to uniquely identify each table.

Advanced synchronization task options for Google BigQuery targets

You can configure advanced task options for a Google BigQuery target on the **Schedule** page of the Synchronization Task wizard.

The following table describes the advanced task options that you can configure for a Google BigQuery target:

Advanced Option	Description
Parameter File Name	Name of the file that contains the definitions and values of user-defined parameters used in the task.
Maximum Number of Log Files	Number of session log files, error log files, and import log files to retain. By default, Data Integration stores each type of log file for 10 runs before it overwrites the log files for new runs.
Update Columns	Specifies the temporary primary key columns to update, upsert, or delete target data. If the Google BigQuery target does not include a primary key column, and the task performs an update, upsert, or delete task operation, click **Add** to add a temporary key. You can select multiple columns. By default, no columns are specified.

Advanced properties for Google BigQuery targets

You can configure advanced target properties on the **Schedule** page of the Synchronization Task wizard.

The following table describes the advanced properties that you can configure for a Google BigQuery target:

Property	Description
Target Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
Target Table Name	Optional. Overrides the Google BigQuery target table name that you specified in the **Target** page of the synchronization task.
Create Disposition	Specifies whether Google BigQuery Connector must create the target table if it does not exist. You can select one of the following values: - Create if needed. If the table does not exist, Google BigQuery Connector creates the table. - Create never. If the table does not exist, Google BigQuery Connector does not create the table and displays an error message.
Write Disposition	Specifies how Google BigQuery Connector must write data in bulk mode if the target table already exists. You can select one of the following values: - Write append. If the target table exists, Google BigQuery Connector appends the data to the existing data in the table. - Write truncate. If the target table exists, Google BigQuery Connector overwrites the existing data in the table. - Write empty. If the target table exists and contains data, Google BigQuery Connector displays an error and does not write the data to the target. Google BigQuery Connector writes the data to the target only if the target table does not contain any data. **Note:** Write disposition is applicable for bulk mode. **Note:** Write disposition is applicable only when you perform an insert operation on a Google BigQuery target.

Property	Description
Write Mode	Specifies the mode to write data to the Google BigQuery target. You can select one of the following modes: - Bulk. In bulk mode, Google BigQuery Connector first writes the data to a staging file in Google Cloud Storage. When the staging file contains all the data, Google BigQuery Connector loads the data from the staging file to the BigQuery target. Google BigQuery Connector then deletes the staging file unless you configure the task to persist the staging file. - Streaming. In streaming mode, Google BigQuery Connector directly writes data to the BigQuery target. Google BigQuery Connector writes the data into the target row by row. Default is Bulk mode.
Streaming Template Table Suffix	Specify the suffix to add to the individual target tables that Google BigQuery Connector creates based on the template target table. This property applies to streaming mode.
Rows per Streaming Request	Specifies the number of rows that Google BigQuery Connector streams to the BigQuery target for each request. Default is 500 rows. The maximum row size that Google BigQuery Connector can stream to the Google BigQuery target for each request is 10 MB. This property applies to streaming mode.
Staging file name	Name of the staging file that Google BigQuery Connector creates in the Google Cloud Storage before it loads the data to the Google BigQuery target. This property applies to bulk mode.
Data Format of the staging file	Specifies the data format of the staging file. You can select one of the following data formats: - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ.
Persist Staging File After Loading	Indicates whether Google BigQuery Connector must persist the staging file in the Google Cloud Storage after it writes the data to the Google BigQuery target. You can persist the staging file if you want to archive the data for future reference. By default, Google BigQuery Connector deletes the staging file in Google Cloud Storage. This property applies to bulk mode.
Enable Staging File Compression	Select this option to compress the size of the staging file before Google BigQuery writes the data to the Google Cloud Storage and decompress the staging file before it loads the data to the Google BigQuery target. You can enable staging file compression to reduce cost and transfer time.
Job Poll Interval in Seconds	The number of seconds after which Google BigQuery Connector polls the status of the write job operation. Default is 10.
Number of Threads for Uploading Staging file	The number of files that Google BigQuery Connector must create to upload the staging file in bulk mode.

Property	Description
Local Stage File Directory	Specifies the directory on your local machine where Google BigQuery Connector stores the files temporarily before writing the data to the staging file in Google Cloud Storage. This property applies to bulk mode.
Allow Quoted Newlines	Indicates whether Google BigQuery Connector must allow the quoted data sections with newline character in a .csv file.
Field Delimiter	Delimiter character for the fields in a .csv file.
Allow Jagged Rows	Indicates whether Google BigQuery Connector must accept the rows without trailing columns in a .csv file.
Pre SQL	SQL statement that you want to run before writing data to the target. For example, if you want to select records from the database before you write the records into the table, specify the following pre SQL statement: `SELECT * FROM `api-project-80697026669.EMPLOYEE.RegionNation` LIMIT 1000`
Pre SQL Configuration	Specify a pre SQL configuration. For example, `DestinationTable:PRESQL_TGT2,DestinationDataset:EMPLOYEE, FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`
Post SQL	SQL statement that you want to run after writing the data into the target. For example, if you want to update records in a table after you write the records into the target table, specify the following post SQL statement: `UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL] SET phoneNumber.number =1000011, phoneNumber.areaCode=100 where fullname='John Doe'`
Post SQL Configuration	Specify a post SQL configuration. For example, `DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE, FlattenResults:True,UseLegacySQL:False`
Success File Directory	Not applicable for Google BigQuery Connector.
Error File Directory	Not applicable for Google BigQuery Connector.

Upsert task operation

When you perform an upsert operation on a Google BigQuery target, you must configure the upsert fields for the target table. You can use an ID field for standard objects. Ensure that you include the upsert field in the field mappings for the task.

Rules and Guidelines

Consider the following rules and guidelines when you perform an upsert operation on a Google BigQuery target:

- You cannot use streaming mode to write data to a Google BigQuery target.
- When you configure a Google BigQuery connection to use simple or hybrid connection mode, you cannot configure upsert operations for columns of the Record data type and repeated columns.
- When you configure a Google BigQuery connection to use complex connection mode, you cannot configure an upsert operation.

CHAPTER 14

Mappings and mapping tasks with Google BigQuery

When you configure a mapping, you describe the flow of data from the source to the target.

A mapping defines reusable data flow logic that you can use in mapping tasks.

When you create a mapping, you define the Source and Target transformations to represent a Google BigQuery object. Use the Mapping Designer in Data Integration to add the Source and Target transformations in the mapping canvas and configure the Google BigQuery Source and Target properties. In advanced mode, the Mapping Designer updates the mapping canvas to include transformations and functions that enable advanced functionality.

You can use Monitor to monitor the jobs.

Pre SQL and post SQL commands

You can specify **pre SQL** and **post SQL** advanced properties for Google BigQuery sources and targets. When you create a task in Data Integration, you can specify SQL commands in the advanced properties for a source and target.

You can perform the following operations by using pre SQL and post SQL commands:

- SELECT
- UPDATE
- DELETE

Note: You cannot perform more than one operation with a pre SQL or post SQL command.

You can configure the options in Google BigQuery with a pre SQL or post SQL statement in the **pre SQL Configuration** or **post SQL Configuration** advanced properties for Google BigQuery sources and targets.

You must use the following format to specify a pre SQL configuration or a post SQL configuration:

```
<Option1:Value1,Option2:Value2,...OptionN:ValueN>
```

The following table shows the configuration options and supported values that you can specify in a pre SQL configuration or post SQL configuration:

Options	Supported Values
DestinationDataset	Dataset ID in Google BigQuery
DestinationTable	Table name in Google BigQuery
FlattenResults	True and False
UseLegacySQL	True and False
WriteDisposition	WRITE_TRUNCATE, WRITE_APPEND, and WRITE_EMPTY

Note: If you perform an UPDATE or DELETE operation with a pre SQL or post SQL command, you must specify the following parameter in the pre SQL configuration or post SQL configuration: `UseLegacySQL:False`

Google BigQuery sources in mappings

To read data from Google BigQuery, configure a Google BigQuery object as the Source transformation in a mapping.

Specify the name and description of the Google BigQuery source. Configure the source, query options, and advanced properties for the source object.

The following table describes the source properties that you can configure for a Google BigQuery source:

Property	Description
Connection	Name of the active Google BigQuery source connection.
Source Type	Type of the Google BigQuery source objects available. You can read data from a single Google BigQuery source object or parameterize the object. You cannot read data from multiple objects.
Object	Name of the Google BigQuery source object based on the source type selected.
Filter	Configure a simple filter or an advanced filter to remove rows at the source. You can improve efficiency by filtering early in the data flow. A simple filter includes a field name, operator, and value. Use an advanced filter to define a more complex filter condition, which can include multiple conditions using the AND or OR logical operators.

The following table describes the advanced properties that you can configure for a Google BigQuery source:

Property	Description
Source Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
Number of Rows to Read	Specifies the number of rows to read from the Google BigQuery source table.
Allow Large Results	Determines whether Google BigQuery Connector must produce arbitrarily large result tables to query large source tables. If you select this option, you must specify a destination table to store the query results.
Query Results Table Name	Required if you select the **Allow Large Results** option. Specifies the destination table name to store the query results. If the table is not present in the dataset, Google BigQuery Connector creates the destination table with the name that you specify.
Job Poll Interval in Seconds	The number of seconds after which Google BigQuery Connector polls the status of the read job operation. Default is 10.
Read Mode	Specifies the read mode to read data from the Google BigQuery source. You can select one the following read modes: - Direct. In direct mode, Google BigQuery Connector reads data directly from the Google BigQuery source table. **Note:** When you use hybrid and complex connection mode, you cannot use direct mode to read data from the Google BigQuery source. - Staging. In staging mode, Google BigQuery Connector exports data from the Google BigQuery source into Google Cloud Storage. After the export is complete, Google BigQuery Connector downloads the data from Google Cloud Storage into the local stage file and then reads data from the local stage file. Default is Direct mode.
Number of Threads for Downloading Staging Files	Specifies the number of files that Google BigQuery Connector downloads at a time to enable parallel download. This property applies to staging mode.
Data Format of the staging file	Specifies the data format of the staging file. You can select one of the following data formats: - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ. **Note:** Avro format is not applicable for Google BigQuery Connector.
Local Stage File Directory	Specifies the directory on your local machine where Google BigQuery Connector stores the Google BigQuery source data temporarily before it reads the data. This property applies to staging mode.
Staging File Name	Name of the staging file where data from the Google BigQuery source table is exported to Google Cloud Storage. This property applies to staging mode.

Property	Description
Enable Staging File Compression	Indicates whether to compress the size of the staging file in Google Cloud Storage before Google BigQuery Connector reads data from the staging file. You can enable staging file compression to reduce cost and transfer time. This property applies to staging mode.
Persist Destination Table	Indicates whether Google BigQuery Connector must persist the query results table after it reads data from the query results table. By default, Google BigQuery Connector deletes the query results table.
pre SQL	SQL statement that you want to run before reading data from the source. For example, if you want to select records in the database before you read the records from the table, specify the following pre SQL statement: `SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;`

pre SQL Configuration	Specify a pre SQL configuration.
	For example,
	`DestinationTable:PRESQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

post SQL	SQL statement that you want to run after reading data from the source.
	For example, if you want to update records in a table after you read the records from a source table, specify the following post SQL statement:
	`UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	`SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where`
	`fullname='John Doe'`

post SQL Configuration	Specify a post SQL configuration.
	For example,
	`DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:True,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

You can set the tracing level in the advanced properties session to determine the amount of details that logs contain.

The following table describes the tracing levels that you can configure:

Property	Description
Terse	The Secure Agent logs initialization information, error messages, and notification of rejected data.
Normal	The Secure Agent logs initialization and status information, errors encountered, and skipped rows due to transformation row errors. Summarizes session results, but not at the level of individual rows.
Verbose Initialization	In addition to normal tracing, the Secure Agent logs additional initialization details, names of index and data files used, and detailed transformation statistics.
Verbose Data	In addition to verbose initialization tracing, the Secure Agent logs each row that passes into the mapping. Also notes where the Secure Agent truncates string data to fit the precision of a column and provides detailed transformation statistics. When you configure the tracing level to verbose data, the Secure Agent writes row data for all rows in a block when it processes a transformation.

Google BigQuery targets in mappings

To write data to a Google BigQuery target, configure a Google BigQuery object as the Target transformation in a mapping.

Specify the name and description of Google BigQuery target. Configure the target and advanced properties for the target object.

The following table describes the target properties that you can configure for a Google BigQuery target:

Property	Description
Connection	Name of the active Google BigQuery connection that is associated with a dataset.
Target Type	Type of the Google BigQuery target objects available. You can write data to a single Google BigQuery target object or parameterize the object. You cannot write data to multiple objects.
Object	Name of the Google BigQuery target object based on the target type selected.
Create New at Runtime	Creates a target. Enter a name for the target object and select the source fields that you want to use. By default, all source fields are used. The target name can contain alphanumeric characters. You can use the following special characters in the file name: ., _, @, $, % Google BigQuery Connector creates a new target table in the **Dataset ID** specified in the Google BigQuery connection.
Operation	You can select one the following operations: - Insert - Update - Upsert (Update or Insert) - Delete **Note:** If you use complex connection mode, you cannot configure update, upsert, and delete operations.
Update Columns	Specifies the temporary primary key columns to update, upsert or delete target data. If the Google BigQuery target does not include a primary key column, and the mapping performs an update, upsert, or delete task operation, click **Add** to add a temporary key. You can select multiple columns. By default, no columns are specified.

The following table describes the advanced properties that you can configure for a Google BigQuery target:

Property	Description
Target Dataset ID	Optional. Overrides the Google BigQuery dataset name that you specified in the connection.
Target Table Name	Optional. Overrides the Google BigQuery target table name that you specified in the **Target** page of the synchronization task.

Property	Description
Create Disposition	Specifies whether Google BigQuery Connector must create the target table if it does not exist. You can select one of the following values: - Create if needed. If the table does not exist, Google BigQuery Connector creates the table. - Create never. If the table does not exist, Google BigQuery Connector does not create the table and displays an error message.
Write Disposition	Specifies how Google BigQuery Connector must write data in bulk mode if the target table already exists. You can select one of the following values: - Write append. If the target table exists, Google BigQuery Connector appends the data to the existing data in the table. - Write truncate. If the target table exists, Google BigQuery Connector overwrites the existing data in the table. - Write empty. If the target table exists and contains data, Google BigQuery Connector displays an error and does not write the data to the target. Google BigQuery Connector writes the data to the target only if the target table does not contain any data. **Note:** Write disposition is applicable for bulk mode. **Note:** Write disposition is applicable only when you perform an insert operation on a Google BigQuery target.
Write Mode	Specifies the mode to write data to the Google BigQuery target. You can select one of the following modes: - Bulk. In bulk mode, Google BigQuery Connector first writes the data to a staging file in Google Cloud Storage. When the staging file contains all the data, Google BigQuery Connector loads the data from the staging file to the BigQuery target. Google BigQuery Connector then deletes the staging file unless you configure the task to persist the staging file. - Streaming. In streaming mode, Google BigQuery Connector directly writes data to the BigQuery target. Google BigQuery Connector writes the data into the target row by row. Default is Bulk mode.
Streaming Template Table Suffix	Specify the suffix to add to the individual target tables that Google BigQuery Connector creates based on the template target table. This property applies to streaming mode.
Rows per Streaming Request	Specifies the number of rows that Google BigQuery Connector streams to the BigQuery target for each request. Default is 500 rows. The maximum row size that Google BigQuery Connector can stream to the Google BigQuery target for each request is 10 MB. This property applies to streaming mode.
Staging file name	Name of the staging file that Google BigQuery Connector creates in the Google Cloud Storage before it loads the data to the Google BigQuery target. This property applies to bulk mode.
Data Format of the staging file	Specifies the data format of the staging file. You can select one of the following data formats: - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. **Note:** In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ.

Property	Description
Persist Staging File After Loading	Indicates whether Google BigQuery Connector must persist the staging file in the Google Cloud Storage after it writes the data to the Google BigQuery target. You can persist the staging file if you want to archive the data for future reference. By default, Google BigQuery Connector deletes the staging file in Google Cloud Storage. This property applies to bulk mode.
Enable Staging File Compression	Select this option to compress the size of the staging file before Google BigQuery writes the data to the Google Cloud Storage and decompress the staging file before it loads the data to the Google BigQuery target. You can enable staging file compression to reduce cost and transfer time.
Job Poll Interval in Seconds	The number of seconds after which Google BigQuery Connector polls the status of the write job operation. Default is 10.
Number of Threads for Uploading Staging file	The number of files that Google BigQuery Connector must create to upload the staging file in bulk mode.
Local Stage File Directory	Specifies the directory on your local machine where Google BigQuery Connector stores the files temporarily before writing the data to the staging file in Google Cloud Storage. This property applies to bulk mode.
Allow Quoted Newlines	Indicates whether Google BigQuery Connector must allow the quoted data sections with newline character in a .csv file.
Field Delimiter	Indicates whether Google BigQuery Connector must allow field separators for the fields in a .csv file.
Allow Jagged Rows	Indicates whether Google BigQuery Connector must accept the rows without trailing columns in a .csv file.
Pre SQL	SQL statement that you want to run before writing data to the target. For example, if you want to select records from the database before you write the records into the table, specify the following pre SQL statement: `SELECT * FROM ``api-project-80697026669.EMPLOYEE.RegionNation`` LIMIT 1000`

Pre SQL Configuration	Specify a pre SQL configuration.
	For example,
	`DestinationTable:PRESQL_TGT2,DestinationDataset:EMPLOYEE,`
	`FlattenResults:False,WriteDisposition:WRITE_TRUNCATE,UseLegacySql:False`

Post SQL	SQL statement that you want to run after writing the data into the target.
	For example, if you want to update records in a table after you write the records into the target table, specify the following post SQL statement:
	`UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL]`
	`SET phoneNumber.number =1000011, phoneNumber.areaCode=100 where`
	`fullname='John Doe'`

Property	Description

Post SQL Configuration	Specify a post SQL configuration.
	For example,
	`DestinationTable:POSTSQL_SRC,DestinationDataset:EMPLOYEE,`
	`FlattenResults:True,UseLegacySQL:False`

Success File Directory	Not applicable for Google BigQuery Connector.
Error File Directory	Not applicable for Google BigQuery Connector.
Forward Rejected Rows	Not applicable for Google BigQuery Connector.

Upsert task operation

When you perform an upsert operation on a Google BigQuery target, you must configure the upsert fields for the target table. You can use an ID field for standard objects. Ensure that you include the upsert field in the field mappings for the task.

Rules and Guidelines

Consider the following rules and guidelines when you perform an upsert operation on a Google BigQuery target:

- You cannot use streaming mode to write data to a Google BigQuery target.
- You cannot configure key range partitioning with more than one partition key.
- When you configure a Google BigQuery connection to use simple or hybrid connection mode, you cannot configure upsert operations for columns of the Record data type and repeated columns.
- When you configure a Google BigQuery connection to use complex connection mode, you cannot configure an upsert operation.

Partitioning

When you read data from a Google BigQuery source and use simple or hybrid connection mode, you can configure key range partitioning to optimize the mapping performance at run time.

Key range partitioning

You can configure key range partitioning when you use a mapping task to read data from Google BigQuery sources and use simple or hybrid connection mode. With key range partitioning, the Secure Agent distributes

rows of source data based on the fields that you define as partition keys. The Secure Agent compares the field value to the range values for each partition and sends rows to the appropriate partitions.

Use key range partitioning for columns that have an even distribution of data values. Otherwise, the partitions might have unequal size. For example, a column might have 10 rows between key values 1 and 1000 and the column might have 999 rows between key values 1001 and 2000. If the mapping includes multiple sources, use the same number of key ranges for each source.

When you define key range partitioning for a column, the Secure Agent reads the rows that are within the specified partition range. For example, if you configure two partitions for a column with the ranges as 10 through 20 and 30 through 40, the Secure Agent does not read the rows 20 through 30 because these rows are not within the specified partition range.

You can configure a partition key for fields of the following data types:

- String
- Integer
- Numeric (only if you use a Google BigQuery connection in hybrid mode)
- Timestamp. Use the following format: YYYY-MM-DD HH24:MI:SS

Note: You cannot configure a partition key for Record data type columns and repeated columns.

You cannot use key range partitions when a mapping includes any of the following transformations:

- Web Services
- JSON to Relational

Configuring Key Range Partitioning

Perform the following steps to configure key range partitioning for Google BigQuery sources:

1. In the Source Properties, click the **Partitions** tab.
2. Select the required partition key from the list.
3. Click **Add New Key Range** to define the number of partitions and the key ranges based on which the Secure Agent must partition data.

   Use a blank value for the start range to indicate the minimum value. Use a blank value for the end range to indicate the maximum value.

   The following image displays the **Partitions** tab:

Hierarchy Parser transformation in mappings

To preserve the hierarchical structure when you read data from Google BigQuery and write data to relational targets, you must use a Hierarchy Parser transformation.

The transformation processes JSON input from the source transformation and provides relational output to the target transformation. The Hierarchy Parser transformation converts hierarchical input based on the sample schema of the Google BigQuery table that you associate with the transformation and the way that you map the data.

Hierarchy Builder transformation in mappings

When you read data from relational sources and write data to a Google BigQuery target, you must use a Hierarchy Builder transformation.

The transformation processes relational input from the upstream transformation and provides JSON output to the downstream transformation. The Hierarchy Builder transformation produces JSON output based on the sample schema of the Google BigQuery table that you associate with the transformation and the way that you map the data.

Rules and Guidelines for mappings and mapping tasks

Consider the following rules and guidelines for mapping and mapping tasks:

- When you write large datasets to a Google BigQuery target, increase the Java heap size in the JVM options for type DTM. Set JVMOption3 to -Xms1024m and JVMOption4 to -Xmx4096m in the **System Configuration Details** section of the Secure Agent and restart the Secure Agent.

- When you use the Hosted Agent as the runtime environment in a mapping task and use a Hierarchy Builder or Hierarchy Parser transformation in a mapping, you must specify a storage path in Google Cloud Storage in the **Schema Definition File Path** field under the connection properties. You can then download the sample schema definition file for the Google BigQuery table from the specified storage path in Google Cloud Storage to a local machine.

- When you read JSON data from a MongoDB source table and write data to a column of Record data type in a Google BigQuery target table, you must specify a explicit value for columns that contain _id in the column name. Otherwise, the task fails with the following error:
  ```
  [ERROR] The [LOAD] job failed with the error - [JSON parsing error in row starting at
  position 0:
  ```

CHAPTER 15

Data type reference

Data Integration uses the following data types in mappings, synchronization tasks, and mapping tasks with Google BigQuery:

Google BigQuery native data types

Google BigQuery data types appear in the **Fields** tab for Source and Target transformations when you choose to edit metadata for the fields.

Transformation data types

Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the Secure Agent uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

When Data Integration reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When Data Integration writes to a target, it converts the transformation data types to the comparable native data types.

Google BigQuery and transformation data types

The following table describes the data types that Data Integration supports for Google BigQuery sources and targets:

Google BigQuery Data Type	Transformation Data Type	Range and Description for the Transformation Data Type
BOOLEAN	String	1 to 104,857,600 characters. Boolean True or False values. Default precision is 5.
DATE	Date/Time	Date values. Google BigQuery Connector uses the following format: DD-MM-YYYY Minimum value: 1/1/1970 Maximum value: 30/12/9999 Precision 29, scale 9

Google BigQuery Data Type	Transformation Data Type	Range and Description for the Transformation Data Type
DATETIME	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond) Default precision 29, scale 9 **Note:** Google BigQuery Connector supports YYYY format for the year.
FLOAT	Double	Precision 15, scale 0
INTEGER	BigInt	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
RECORD	String	1 to 104,857,600 characters Default precision is 255. You can increase the value up to 104857600 characters.
NUMERIC	Decimal	Default precision 28, scale 10 **Note:** Though the Fields tab shows the scale as 10, Google BigQuery supports scale upto 9.
STRING	String	1 to 104,857,600 characters Default precision is 255. You can increase the value up to 104857600 characters.
BYTE	Byte	1 to 104,857,600 bytes
TIME	Date/Time	Time values. (precision to the nanosecond) Precision 29, scale 9
TIMESTAMP	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond) Precision 29, scale 9 **Note:** Google BigQuery Connector supports YYYY format for the year.

Note: When you write data of the Timestamp data type from a Google BigQuery source object, certain Timestamp values written to the target show a difference by a few milliseconds from the value in the source.

INDEX