



Informatica® Data Integration - Free & PayGo

Google BigQuery V2 Connector

© Copyright Informatica LLC 2018, 2023

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, the Informatica logo, Informatica Cloud, and PowerCenter are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

See patents at <https://www.informatica.com/legal/patents.html>.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2023-04-04

Table of Contents

| | |
|---|-----------|
| Preface | 5 |
| Informatica Resources. | 5 |
| Informatica Documentation. | 5 |
| Informatica Intelligent Cloud Services web site. | 5 |
| Informatica Intelligent Cloud Services Communities. | 5 |
| Informatica Intelligent Cloud Services Marketplace. | 6 |
| Data Integration connector documentation. | 6 |
| Informatica Knowledge Base. | 6 |
| Informatica Intelligent Cloud Services Trust Center. | 6 |
| Informatica Global Customer Support. | 6 |
| Chapter 1: Introduction to Google BigQuery V2 Connector..... | 7 |
| Google BigQuery V2 Connector assets. | 7 |
| Google BigQuery example. | 7 |
| Administration of Google BigQuery V2 Connector. | 8 |
| Chapter 2: Google BigQuery V2 connections..... | 9 |
| Connection modes. | 9 |
| Connection mode example. | 9 |
| Rules and guidelines for Google BigQuery V2 connection modes. | 12 |
| Google BigQuery V2 connection properties. | 14 |
| Configure proxy settings. | 16 |
| Configure proxy settings for NTLM authentication. | 17 |
| Chapter 3: Google BigQuery pushdown optimization..... | 18 |
| Pushdown optimization types. | 18 |
| Pushdown optimization scenarios. | 19 |
| Pushdown optimization preview. | 20 |
| Pushdown optimization using a Google BigQuery V2 connection. | 20 |
| Read from and write to Google BigQuery. | 21 |
| Read from Google Cloud Storage and write to Google BigQuery. | 21 |
| Read from Amazon S3 and write to Google BigQuery. | 21 |
| Configuring pushdown optimization for a Google BigQuery mapping task. | 22 |
| Pushdown compatibility. | 22 |
| Configuring a custom query or an SQL override for the Google BigQuery source object. | 32 |
| Cancelling a pushdown optimization task. | 32 |
| Clean stop a pushdown optimization job. | 32 |
| Context based optimization for multiple targets. | 33 |
| Rules and guidelines for pushdown optimization. | 34 |
| Set partial pushdown processing to false. | 41 |

| | |
|--|---------------|
| Chapter 4: Mappings for Google BigQuery V2..... | 42 |
| Custom query source type. | 42 |
| Read modes. | 43 |
| Optimize read performance in staging mode. | 43 |
| Write modes. | 45 |
| Optimize the staging performance | 45 |
| Data filters. | 47 |
| Pre-SQL and post-SQL commands. | 47 |
| Adding multiple source objects. | 48 |
| Rules and guidelines for adding multiple source objects. | 50 |
| Handling dynamic schemas. | 51 |
| Rules and guidelines for dynamic schema handling. | 52 |
| Google BigQuery V2 sources in mappings. | 52 |
| Google BigQuery V2 lookups in mappings | 56 |
| Unconnected lookup transformation. | 60 |
| Configuring an unconnected lookup transformation. | 60 |
| Enabling lookup caching. | 63 |
| Google BigQuery V2 targets in mappings. | 63 |
| Mapping tasks with CDC sources. | 69 |
| Upsert task operation. | 71 |
| Using Merge query for update, upsert, and delete operations. | 71 |
| Data driven operation for mappings. | 71 |
| Determine the order of processing for multiple targets. | 72 |
| Process SQL queries using an SQL transformation. | 73 |
| Configuring an SQL transformation. | 74 |
| Using a parameterized connection in an SQL transformation. | 75 |
| Rules and guidelines for SQL transformation. | 75 |
| Configure unique staging object names for concurrent mappings. | 76 |
| Rules and Guidelines for mapping and mapping tasks. | 78 |
| Troubleshooting a mapping task. | 82 |
| Appendix A: Data type reference..... | 83 |
| Google BigQuery V2 and transformation data types. | 83 |
| Index. | 86 |

Preface

Use *Google BigQuery V2 Connector* to learn how to read from or write to Google BigQuery by using Data Integration. Learn to create a connection, develop mappings, and run mapping tasks in Data Integration. You can also learn how to configure pushdown optimization to a Google BigQuery database using a Google BigQuery ODBC connection.

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 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 Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at <http://www.informatica.com/cloud>. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

<https://network.informatica.com/community/informatica-network/products/cloud-integration>

Developers can learn more and share tips at the Cloud Developer community:

<https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers>

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and maplets:

<https://marketplace.informatica.com/>

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit <https://docs.informatica.com>.

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 Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at <https://www.informatica.com/trust-center.html>.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The [Informatica Intelligent Cloud Services Status](#) page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, go to <https://status.informatica.com/> and click **SUBSCRIBE TO UPDATES**. You can then choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or online.

For online support, click **Submit Support Request** in Informatica Intelligent Cloud Services. You can also use Online Support to log a case. Online Support requires a login. You can request a login at <https://network.informatica.com/welcome>.

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>.

CHAPTER 1

Introduction to Google BigQuery V2 Connector

You can use Google BigQuery V2 Connector to securely read data from or write data to Google BigQuery.

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.

When you use Google BigQuery V2 Connector 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 V2 connection configuration.

You can move data from any data source to Google BigQuery.

Use Google BigQuery V2 Connector to create a mapping, data transfer task, or a mapping task.

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
- 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

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 a Google service account to access Google BigQuery.
- Ensure you have the `client_email`, `project_id`, and `private_key` values for the service account. You will need to enter these details when you create a Google BigQuery V2 connection in Data Integration.
- 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.
- When you only read data from a Google BigQuery table, you must have the required permissions to read data 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.

CHAPTER 2

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.

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.

Connection mode example

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:

Field map options: Automatic Note: This option will automatically map any fields added later by name. Options

Incoming Fields: (112 of 112 mapped)

Find

| Field Name |
|---|
| Fld_String |
| Fld_Integer |
| Fld_Float |
| Fld_Date |
| Fld_Time |
| Fld_DateTime |
| Fld_Timestamp |
| Fld_Boolean |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Boolean |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Timestamp |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_DateTime |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Time |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Date |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Float |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Integer |

Target Fields: (112 of 112 mapped)

Find

| Field Name | Mapped Field |
|---|---|
| Fld_String | Fld_String |
| Fld_Integer | Fld_Integer |
| Fld_Float | Fld_Float |
| Fld_Date | Fld_Date |
| Fld_Time | Fld_Time |
| Fld_DateTime | Fld_DateTime |
| Fld_Timestamp | Fld_Timestamp |
| Fld_Boolean | Fld_Boolean |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Boolean | Master1_Nullable_Level1_Repeated_Nullable_Fld_Boolean |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Timestamp | Master1_Nullable_Level1_Repeated_Nullable_Fld_Timestamp |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_DateTime | Master1_Nullable_Level1_Repeated_Nullable_Fld_DateTime |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Time | Master1_Nullable_Level1_Repeated_Nullable_Fld_Time |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Date | Master1_Nullable_Level1_Repeated_Nullable_Fld_Date |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Float | Master1_Nullable_Level1_Repeated_Nullable_Fld_Float |
| Master1_Nullable_Level1_Repeated_Nullable_Fld_Integer | Master1_Nullable_Level1_Repeated_Nullable_Fld_Integer |

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:

Field map options: Automatic Note: This option will automatically map any fields added later by name. Options

Incoming Fields: (20 of 20 mapped) Find

| Field Name ^ |
|---------------------------|
| Fld_String |
| Fld_Integer |
| Fld_Float |
| Fld_Date |
| Fld_Time |
| Fld_DateTime |
| Fld_Timestamp |
| Fld_Boolean |
| Master1_Nullable |
| Master2_Repeated |
| Master3_Nullable_Repeated |
| Master4_Repeated_Nullable |
| Fld_String_Repeated |
| Fld_Integer_Repeated |
| Fld_Float_Repeated |

Target Fields: (20 of 20 mapped) Find

| Field Name ^ | Mapped Field |
|---------------------------|---------------------------|
| Fld_String | Fld_String |
| Fld_Integer | Fld_Integer |
| Fld_Float | Fld_Float |
| Fld_Date | Fld_Date |
| Fld_Time | Fld_Time |
| Fld_DateTime | Fld_DateTime |
| Fld_Timestamp | Fld_Timestamp |
| Fld_Boolean | Fld_Boolean |
| Master1_Nullable | Master1_Nullable |
| Master2_Repeated | Master2_Repeated |
| Master3_Nullable_Repeated | Master3_Nullable_Repeated |
| Master4_Repeated_Nullable | Master4_Repeated_Nullable |
| Fld_String_Repeated | Fld_String_Repeated |
| Fld_Integer_Repeated | Fld_Integer_Repeated |
| Fld_Float_Repeated | Fld_Float_Repeated |

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:

Field map options: Automatic Note: This option will automatically map any fields added later by name. Options

Incoming Fields: (1 of 1 mapped) Find

| Field Name ^ |
|--------------|
| STRING_DATA |

Target Fields: (1 of 1 mapped) Find

| Field Name ^ | Mapped Field |
|--------------|--------------|
| STRING_DATA | STRING_DATA |

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 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 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

Google BigQuery V2 connection properties

When you create a Google BigQuery V2 connection, configure the connection properties.

The following table describes the Google BigQuery V2 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. |
| Type | The Google BigQuery V2 connection type. |
| Runtime Environment | Name of the runtime environment where you want to run the tasks. Select a Secure Agent or a Hosted Agent. |
| Service Account ID | The client_email value in the JSON file that you download after you create a service account. |
| Service Account Key | The private_key value in the JSON file that you download after you create a service account. |
| Project ID | The project_id value 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. |
| Storage Path | Path in Google Cloud Storage where the agent creates a local stage file to store the data temporarily. Applies to tasks that read or write large volumes of data. Use this property when you read data in staging mode or write data in bulk mode. You can either enter the bucket name or the bucket name and folder name. Use one of the following formats: - gs://<bucket name> - gs://<bucket name>/<folder_name> |
| 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 V2 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 | <p>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.</p> <p>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.</p> <p>The schema definition file is required if you configure complex connection mode in the following scenarios:</p> <ul style="list-style-type: none"> - 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. |
| Use Legacy SQL For Custom Query | Select this option to use a legacy SQL to define a custom query. If you clear this option, you must use a standard SQL to define a custom query. |
| Dataset Name for Custom Query | When you define a custom query, you must specify a Google BigQuery dataset. |
| Region Id | <p>The region name where the Google BigQuery dataset that you want to access resides.</p> <p>Note: You must 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.</p> <p>For more information about the regions supported by Google BigQuery, see Dataset locations.</p> |
| Staging Dataset | 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. |
| Optional Properties | <p>Specifies whether you can configure source and target functionality through custom properties.</p> <p>You can select one of the following options:</p> <ul style="list-style-type: none"> - None. If you do not want to configure any custom properties, select None. - Required. If you want to specify custom properties to configure the source and target functionalities. <p>Default is None.</p> |
| Provide Optional Properties | <p>Comma-separated key-value pairs of custom properties in the Google BigQuery V2 connection to configure certain source and target functionalities.</p> <p>Appears when you select Required in the Optional Properties.</p> <p>For more information about the list of custom properties that you can specify, see the Informatica Knowledge Base article: https://kb.informatica.com/faq/7/Pages/26/632722.aspx</p> |

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.

Retry Strategy

When you read data from Google BigQuery in staging mode and write data to Google BigQuery in bulk mode, you can configure the retry strategy when the Google BigQuery V2 connection fails to connect to the Google BigQuery. The retry strategy is not applicable in the CDC and streaming modes when you write data to a Google BigQuery target.

The following table describes the retry properties for the Google BigQuery V2 connection. These retry properties apply only to mappings.

| Property | Description |
|------------------------|--|
| Enable Retry | Indicates that the Secure Agent attempts to retry the connection when there is a failure. Select this option to enable connection retry. Default is unselected. |
| Maximum Retry Attempts | 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. Appears when you select the Enable Retry property. |
| Initial Retry Delay | The initial wait time in seconds before the Secure Agent attempts to retry the connection. Default is 1. Appears when you select the Enable Retry property. |
| Retry Delay Multiplier | 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 is 2.0. Appears when you select the Enable Retry property. |
| Maximum Retry Delay | The maximum wait time in seconds that the Secure Agent waits between successive retry attempts. Default is 32. Appears when you select the Enable Retry property. |
| Total Timeout | The total time duration in seconds that the Secure Agent attempts to retry the connection after which the connection fails. Default is 50. Appears when you select the Enable Retry property. |

Configure proxy settings

If your organization uses an outgoing proxy server to connect to the internet, the agent connects to Informatica Intelligent Cloud Services through the proxy server.

You can configure the Secure Agent to use the proxy server on Windows and Linux. You can use the unauthenticated or authenticated proxy server.

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 the topic "Configure the proxy settings on Windows" or "Configure the proxy settings on Linux," in *Getting Started* in the Data Integration documentation.
- Configure the JVM options for the DTM in the Secure Agent properties. For instructions, see the Knowledge Base article [000185646](#).

Contact your network administrator for the correct proxy settings.

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 3

Google BigQuery pushdown optimization

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

When you run a task configured for pushdown 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 pushdown optimization for a mapping in the tasks properties. Full pushdown optimization is enabled by default in mapping tasks.

Pushdown optimization types

When you apply pushdown 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 pushdown 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 pushdown 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 pushdown optimization, Data Integration pushes the transformation logic for all the supported transformations downstream in the mapping, but excludes the target transformation.

Pushdown optimization scenarios

You can configure pushdown optimization using the Google BigQuery V2 Connector or the Google BigQuery ODBC Connector in mappings.

Google BigQuery V2 Connector

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

You can use configure pushdown optimization for the following scenarios when you use Google BigQuery V2 Connector in mappings:

| Source and target endpoints | Supported pushdown scenarios in mappings | Pushdown optimization type |
|---|---|--|
| Google BigQuery source Google BigQuery target | Reads from and writes to Google BigQuery using the Google BigQuery V2 connection. | Full, Source Note: The Secure Agent pushes only partial or the entire mapping logic for processing to Google BigQuery. |
| 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 pushdown optimization. | | |

ODBC Connector

Important: Informatica recommends that you use the Google BigQuery V2 connection in mappings to configure pushdown optimization. If you cannot push down specific transformation logic using the Google BigQuery V2 connection, you can explore configuring pushdown optimization using the Google BigQuery ODBC connection.

You can configure pushdown for a mapping that uses a Google BigQuery ODBC connection to read from and write to Google BigQuery.

| Source and target endpoints | Supported pushdown scenarios in mappings | Pushdown optimization type |
|---|---|----------------------------|
| Google BigQuery source Google BigQuery target | Reads from and writes to Google BigQuery using the Google BigQuery ODBC connection with the Google BigQuery ODBC subtype. The Google BigQuery ODBC connection uses the Google BigQuery ODBC 64-bit drivers on Windows and Linux systems. You must configure the Google BigQuery driver based on your system requirement. | Full, Source |
| Note: You can use the Secure Agent to run mappings enabled with pushdown optimization. | | |

Pushdown optimization preview

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

After you select the required pushdown optimization options and run the preview, Data Integration creates and runs a temporary pushdown preview mapping task. When the job completes, Data Integration displays the SQL queries to be executed and any warnings in the **Pushdown Optimization** panel. The warning messages help you understand which transformations in the configured mapping are not applicable for pushdown optimization. If pushdown 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 pushdown 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 pushdown optimization, see the topic "Pushdown optimization preview" in *Mappings* in the Data Integration help.

Pushdown optimization using a Google BigQuery V2 connection

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

You can configure pushdown 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.

Read from and write to Google BigQuery

You can configure pushdown 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 pushdown optimization in the mapping to enhance the performance.

Read from Google Cloud Storage and write to Google BigQuery

You can configure pushdown 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 pushdown optimization to write data to the Google BigQuery target to enhance the performance and reduce the cost involved.

Read from Amazon S3 and write to Google BigQuery

You can configure pushdown 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 pushdown optimization in the mapping to optimize the performance.

Configuring pushdown optimization for a Google BigQuery mapping task

Perform the following steps to configure pushdown optimization for an Google BigQuery V2 mapping task:

1. Create an Google Cloud Storage V2 connection and an Google BigQuery V2 connection.
2. Create a mapping to read data from an Google Cloud Storage source and write data to an Google BigQuery target.
3. Create a mapping task.
 - a. Select the configured mapping.
 - b. In the **Pushdown Optimization** section on the **Schedule** tab, set the pushdown optimization value to **Full**.
 - c. Save the task and click **Finish**.

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

Pushdown compatibility

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

When you use pushdown 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 for Google BigQuery V2 mappings

The following table summarizes the availability of pushdown functions in an Google BigQuery database. Columns marked with an X indicate that the function can be pushed to the Google BigQuery database by using full pushdown optimization. Columns marked with a dash (-) symbol indicate that the function cannot be pushed to the database.

| Function | Pushdown | Function | Pushdown | Function | Pushdown |
|---------------|----------|-------------|----------|-----------------|----------|
| ABORT() | - | INSTR() | X | REPLACECHR() | X |
| ABS() | X | IS_DATE() | X | REPLACESTR() | X |
| ADD_TO_DATE() | X | IS_NUMBER() | X | REVERSE() | - |
| AES_DECRYPT() | - | IS_SPACES() | X | ROUND(DATE) | X |
| AES_ENCRYPT() | - | ISNULL() | X | ROUND(NUMBER) | X |
| ASCII() | - | LAST() | - | RPAD() | X |
| AVG() | x | LAST_DAY() | X | RTRIM() | X |
| CEIL() | X | LEAST() | - | SET_DATE_PART() | - |
| CHOOSE() | - | LENGTH() | X | SIGN() | X |
| CHR() | X | LN() | X | SIN() | X |

| Function | Pushdown | Function | Pushdown | Function | Pushdown |
|-----------------|----------|------------------|----------|-----------------|----------|
| CHRCODE() | - | LOG() | X | SINH() | X |
| COMPRESS() | - | LOOKUP | - | SHA256() | - |
| CONCAT() | X | LOWER() | X | SOUNDEX() | - |
| COS() | X | LPAD() | X | SQRT() | X |
| COSH() | X | LTRIM() | X | STDDEV() | X |
| COUNT() | X | MAKE_DATE_TIME() | - | SUBSTR() | X |
| CRC32() | - | MAX() | X | SUM() | X |
| CUME() | - | MD5() | - | SYSDATE() | X |
| DATE_COMPARE() | - | MEDIAN() | - | SYSTIMESTAMP() | X |
| DATE_DIFF() | - | METAPHONE() | - | TAN() | X |
| DECODE() | X | MIN() | X | TANH() | - |
| DECODE_BASE64() | - | MOD() | X | TO_BIGINT | X |
| DECOMPRESS() | - | MOVINGAVG() | - | TO_CHAR(DATE) | X |
| ENCODE_BASE64() | - | MOVINGSUM() | - | TO_CHAR(NUMBER) | X |
| EXP() | X | NPER() | - | TO_DATE() | X |
| FIRST() | - | PERCENTILE() | - | TO_DECIMAL() | X |
| FLOOR() | X | PMT() | - | TO_FLOAT() | X |
| FV() | - | POWER() | X | TO_INTEGER() | X |
| GET_DATE_PART() | X | PV() | - | TRUNC(DATE) | X |
| GREATEST() | - | RAND() | - | TRUNC(NUMBER) | X |
| IIF() | X | RATE() | - | UPPER() | X |
| IN() | - | REG_EXTRACT() | - | VARIANCE() | X |
| INDEXOF() | - | REG_MATCH() | - | | |
| INITCAP() | - | REG_REPLACE | - | | |

Supported operators for Google BigQuery V2 mappings

When you use pushdown 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 pushdown operators that can be used in an Google BigQuery database. Columns marked with an X indicate that the operator can be pushed to the Google BigQuery database by using full pushdown optimization.

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

Transformations for Google BigQuery V2 mappings

When you configure pushdown 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:

- Aggregator¹
- Expression
- Filter
- Joiner¹
- Lookup¹
- Rank¹
- Router¹
- Sequence Generator
- Sorter¹
- SQL
- Union¹
- Update Strategy¹

¹Applicable only for Google BigQuery sources.

Note: Sequence Generator and Update Strategy transformations can't be pushed to the database and all other transformations can be pushed to Google BigQuery by using full pushdown optimization.

Aggregator transformation

You can configure full pushdown 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 pushdown 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 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 pushdown 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 pushdown 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 pushdown 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 pushdown 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
 - Inout and input parameters in a query

Supported variables for Google BigQuery V2 mappings

When you use pushdown 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 pushdown 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 pushdown optimization.

| Variable | Pushdown |
|-------------------|----------|
| SESSSTARTTIME | X |
| SYSDATE | - |
| WORKFLOWSTARTTIME | - |

Supported data types for Google BigQuery V2 mappings

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 | <ul style="list-style-type: none">- BigInt- Number- String |
| Avro | <ul style="list-style-type: none">- Binary- Byte- Double- Float- Int- Long- String |
| Parquet | <ul style="list-style-type: none">- Binary- Date- Decimal- Double- Float- Int32- Int64- Int96- String |
| JSON | <ul style="list-style-type: none">- Double- Int- Long- String |

The following table lists the Google BigQuery data types that can be used for pushdown 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 |
| Byte | Byte |

| Google BigQuery Data Type | Transformation Data Type |
|---------------------------|--------------------------|
| Time | Date/Time |
| Timestamp | Date/Time |

Supported features for Google BigQuery V2 mappings

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

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

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

- Source type - Single
- Source type - Query
- Source type - Standard and materialized views
- Source Object Type - 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 pushdown 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 pushdown optimization, the mappings support the following properties for an Google BigQuery V2 target:

- Target Object Type - Single
- Target Object Type - Parameter
- Operation
 - Insert
 - Update
 - Upsert
 - Delete
 - Data Driven
- 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 pushdown.

[Guidelines for mappings that read from and write to Google BigQuery](#)

When you configure pushdown 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 pushdown 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 pushdown optimization fails.

Supported features for Google Cloud Storage V2 source

When you configure pushdown optimization, the Google Cloud Storage V2 connection supports the following properties:

- Service Account ID
- Service Account Key
- Project ID

When you configure pushdown 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.
- Delimited file formatting options
 - Delimiter
 - Qualifier
 - Header Line Number
 - First Data Row
- Google Cloud Storage Path
- Source File Name
- Is Directory

Supported features for Amazon S3 source

When you configure pushdown optimization, the following connection properties of Amazon S3 V2 source are supported:

- Access Key
- Secret Key

- Folder Path

When you configure pushdown 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 pushdown 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 pushdown optimization in a mapping that reads from an Amazon S3 source and writes to a Google BigQuery target:

- 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.
- 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 pushdown optimization.
- Mapping fails when you read boolean values from an Amazon S3 source and write 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 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.
- 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.
- 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 pushdown optimization.

- When you use a smaller dataset (TPCH scale factor 1 or below) in a mapping and enable pushdown optimization, the mapping requires 30% more time to process the data when compared with the same mapping without pushdown optimization.
- When you configure a mapping enabled with pushdown optimization to read a boolean integer column and write to a boolean string column, the mapping fails.

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 pushdown optimization.

Perform the following task to set the property:

1. In the mapping task, navigate to the **Pushdown Optimization** section on the **Schedule** tab.
2. Select **Create Temporary View**.
3. Click **Finish**.

Cancelling a pushdown optimization task

When you configure full pushdown optimization for a task that reads from and writes to Google BigQuery, you can determine how Data Integration handles the job when pushdown optimization does not work.

You can set the task to fail or run without pushdown optimization. You can use this functionality only when you configure full pushdown optimization for a task that runs using the Google BigQuery V2 connection in the Source and Target transformations.

1. Create a mapping task.
2. Select the configured mapping for which you want to enable pushdown optimization.
3. In the **Pushdown Optimization** section on the **Schedule** tab, set the pushdown optimization value to **Full** for the selected mapping.
4. To determine what action Data Integration must perform when pushdown optimization does not occur, enable or disable the **If pushdown optimization is not possible, cancel the task** option based on your requirement:
 - a. Enable to cancel the task when pushdown does not occur.
 - b. Disable to run the task without pushdown optimization.
Default is disabled.
5. Save the task and click **Finish**.

Clean stop a pushdown optimization job

When a task enabled for pushdown 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 pushdown 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 pushdown optimization task:

- When you clean stop a task enabled for source pushdown 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 pushdown 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.

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 pushdown 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 pushdown 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.

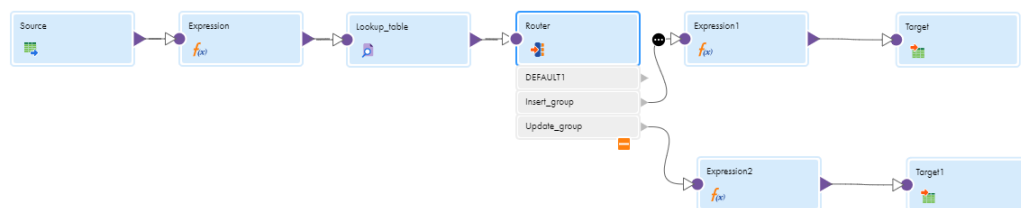
You can enable the **SCD Type 2 merge** when you write to same Google BigQuery table in two Target transformations and perform different operations for each of the Target transformations, where you use one target to insert data and the other target to update data. Data Integration combines the queries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

Rules and guidelines for pushdown optimization

Certain rules and guidelines apply when you enable a mapping for pushdown optimization to a Google BigQuery database.

Sources and targets

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 pushdown optimization for a mapping task when the target table contains columns of record data type or repeated columns.
- You cannot enable full pushdown 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 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, pushdown query fails.
- 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 pushdown 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 pushdown.

- 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 pushdown.
- 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 pushdown 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 withing a transformation and one of the functions cannot be pushed to Google BigQuery, the mapping runs without pushdown 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 pushdown 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 pushdown 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.

Lookup transformation

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 pushdown, 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 pushdown 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.

Functions

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 pushdown optimization, the IS_DATE() returns Boolean values of 0 or 1 to the Google BigQuery target table. If the mappings run with pushdown 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 pushdown 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 pushdown optimization and a mapping that runs without pushdown 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 pushdown or with partial pushdown 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 pushdown optimization.
- When you push the SYSDATE() function to the Google BigQuery database and the mapping runs with pushdown 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 pushdown 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 pushdown 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 pushdown 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 pushdown optimization and a mapping that runs without pushdown 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 pushdown 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 pushdown.
- 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 pushdown optimization against a mapping disabled for pushdown optimization.

If the mapping runs with full pushdown optimization, the trailing zeroes after decimal is not considered while casting. However, if you run a mapping with disabled pushdown 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 pushdown optimization.

You might encounter a data mismatch when you compare a mapping that runs with full pushdown optimization and a mapping that runs without pushdown optimization. The microseconds are not considered in the mapping that runs without pushdown 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 pushdown optimization and a mapping that runs without pushdown 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 pushdown optimization, the default date format of the data/time data types returned for a mapping with pushdown optimization is YYYY-MM-DD HH24:MI:SS.US, whereas for a mapping without pushdown optimization is MM/DD/YYYY HH24:MI:SS.US.

To fix the issue in a mapping without pushdown 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 YYYY-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 `$$$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.

Set partial pushdown processing to false

If you enable a mapping for full pushdown optimization and the mapping contains unsupported transformations or functions, the mapping is partially processed.

If you do not want the mapping to be partially processed and run the mapping without pushdown optimization, you can set the following property in the Google BigQuery V2 connection optional properties:
DisablePDOIfPartiallyPushed:true

CHAPTER 4

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.

You can use Monitor to monitor the jobs.

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 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, click the following URL:

<https://cloud.google.com/bigquery/docs/reference/legacy-sql>

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:
 - 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;`

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 then processes this data read.

Consider the following rules when you enable the staging property:

- If you run a mapping enabled for pushdown optimization, the mapping does not consider the staging property and runs with pushdown optimization.
- When you read data of the byte data type from the Google BigQuery source, ensure that size or precision of the binary data does not exceed more than 62914560 bytes.
- Ensure that the total size or precision of all the columns in the Google BigQuery source does not exceed more than 125829120 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.
- If you run a mapping enabled for staging optimization with Google BigQuery object type and the advanced fields are parameterized, and the **Allow Parameter to be overridden at run time** option is selected in the input parameter window while adding the parameters, the mapping does not consider the staging property and runs without staging optimization.

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 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 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].
```

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 the staging performance

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.

Consider the following rules when you enable the staging property:

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

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:



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.

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, you must ensure that you use a Google BigQuery V2 connection in hybrid connection 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:

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

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

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

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:

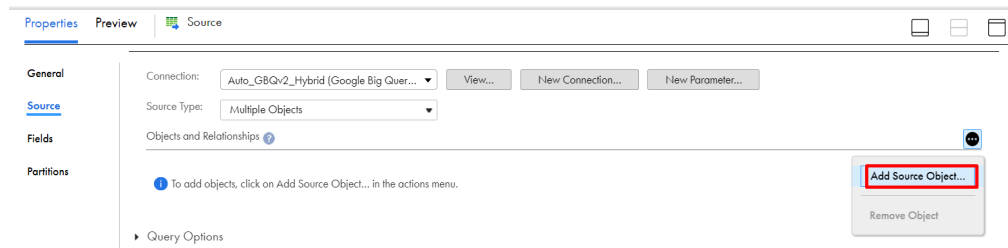
| Options | Supported Values |
|--------------------|---|
| DestinationDataset | Dataset ID in Google BigQuery |
| DestinationTable | Table name in Google BigQuery |
| FlattenResults | True and False |
| UseLegacySQL | True and False |
| WriteDisposition | WRITE_TRUNCATE, WRITE_APPEND, and WRITE_EMPTY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

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`

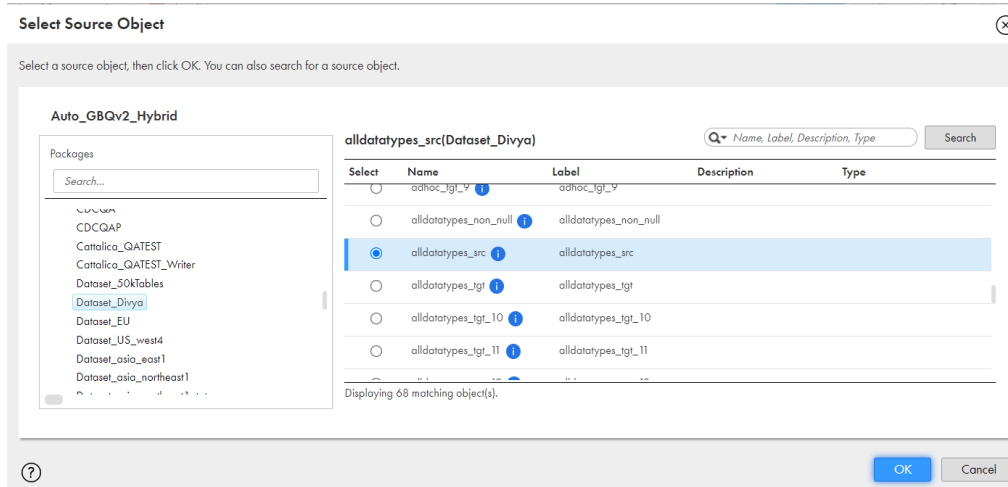
Adding multiple source objects

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.

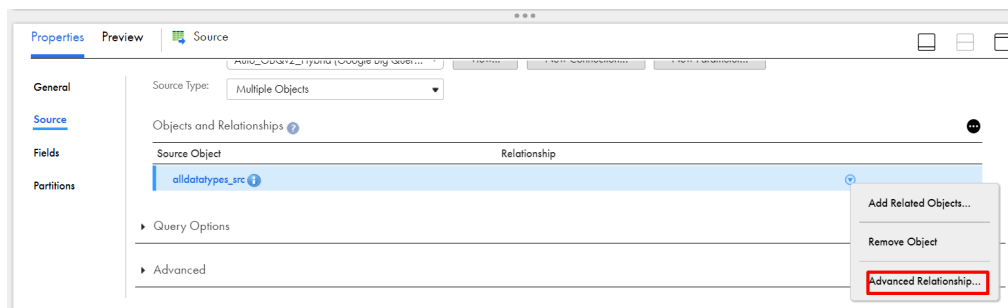
1. In the Source transformation, click the **Source Type** as **Multiple Objects**.
2. From the **Actions** menu, click **Add Source Object**.



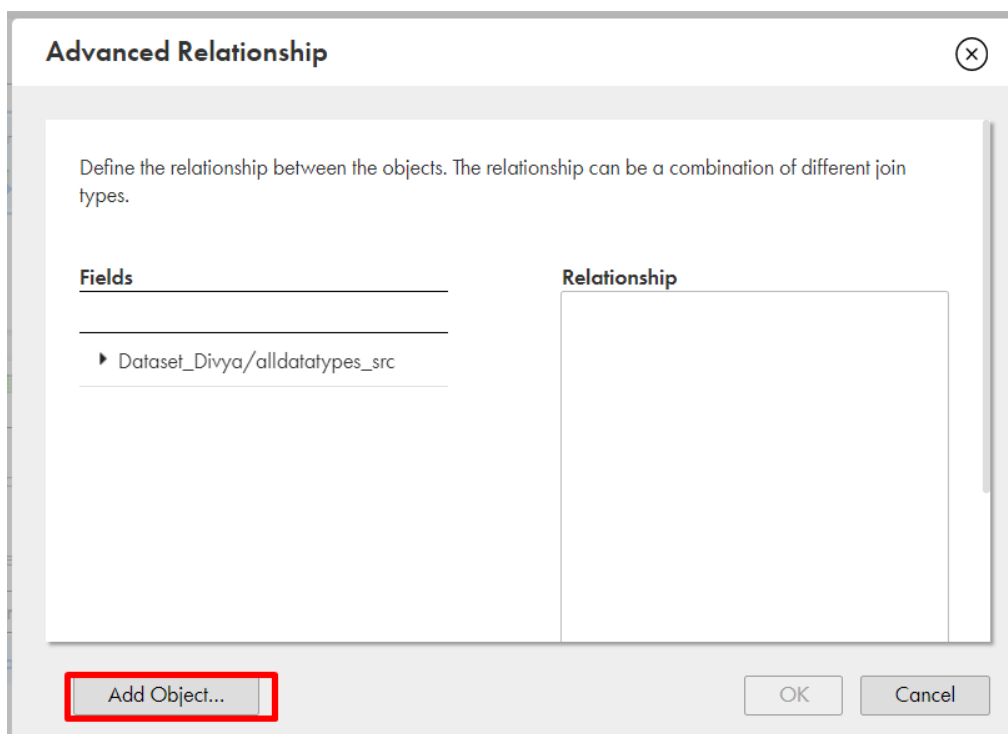
3. Select the source object that you want to add from the displayed list.



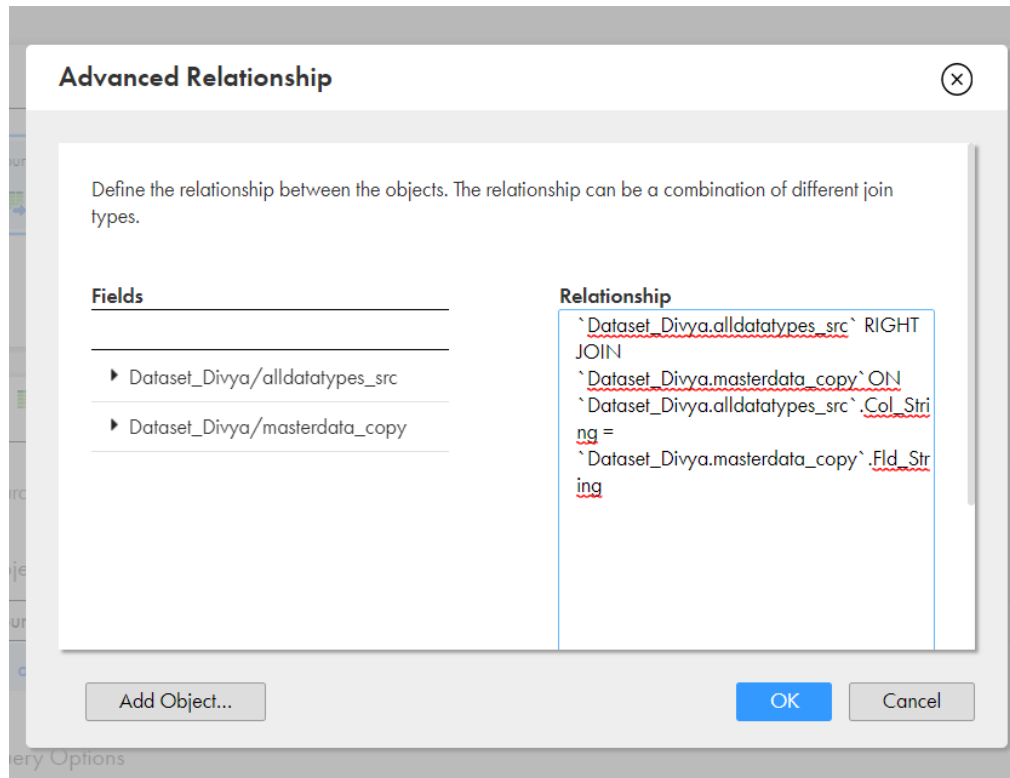
- Click **OK**.
- From the **Related Objects Actions** menu, select **Advanced Relationship**:



- In the **Advanced Relationship** window, you can click **Add Object** to add more objects.



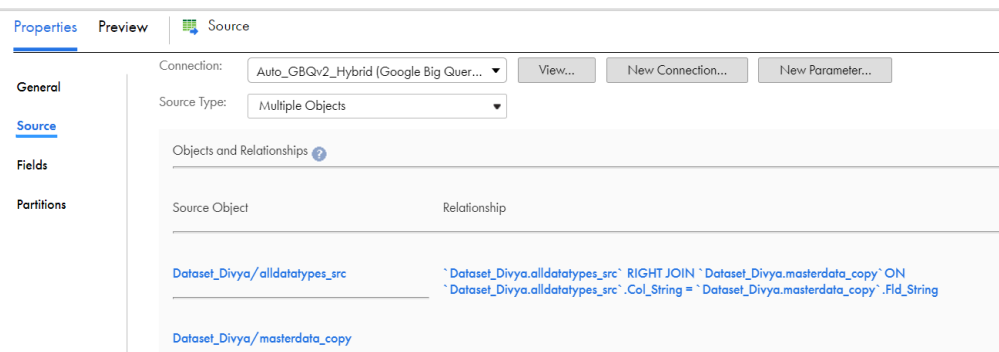
7. Click **OK**.
8. Set your own conditions or specify a query to define the relationship between the tables:



Note: When you configure a join expression, select the fields and define a join condition or a query syntax.

9. 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 must not import multiple source tables with the same name from different datasets.

- When you specify a cross join condition to read data from multiple Google BigQuery tables, use the following format:
`datasetname.tablename CROSS JOIN datasetname.tablename`
- You cannot use a self join when you add multiple source tables.
- 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.
- When you specify a SQL Override query for multiple source tables, use the following format for the SQL override query:
`select `dataset.table`.col1 AliasPrefix_dataset_table_col1, `dataset.table`.col2
AliasPrefix_dataset_table_col2, `dataset1.table1`.col1
AliasPrefix_dataset1_table1_col1, `dataset1.table1`.col2 AliasPrefix_dataset1_table1_col2
from `dataset.table` <join condition> `dataset1.table1` ON `dataset.table`.col =
`dataset1.table1`.col1 where <condition>`

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 | <p>Data Integration refreshes the schema every time the task runs.</p> <p>You can choose from the following options to refresh the schema:</p> <ul style="list-style-type: none">- Alter and apply changes. Data Integration applies the following changes from the source schema to the target schema:<ul style="list-style-type: none">- 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.

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 | <p>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.</p> <p>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.</p> |
| Source Type | <p>Type of the Google BigQuery source objects available.</p> <p>Select Single Object, Multiple, Query or Parameter.</p> |
| Object | <p>Name of the Google BigQuery source object based on the source type selected.</p> |

| Property | Description |
|-----------|--|
| Parameter | <p>A parameter file where you define values that you want to update without having to edit the task.</p> <p>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.</p> <p>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.</p> |
| Query | <p>Click on Define Query and enter a valid custom query.</p> <p>The Query property appears only if you select Query as the source type.</p> <p>You can parameterize a custom query object at runtime in a mapping.</p> <p>Select the source advanced property Use EXPORT DATA Statement to stage to use the ORDER BY clause in a custom query in staging mode.</p> |
| Filter | <p>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.</p> <p>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.</p> |

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 | 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 | <p>Determines whether Google BigQuery V2 Connector must produce arbitrarily large result tables to query large source tables.</p> <p>If you select this option, you must specify a destination table to store the query results.</p> |
| Query Results Table Name | <p>Required if you select the Allow Large Results option.</p> <p>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.</p> |
| Job Poll Interval in Seconds | <p>The number of seconds after which Google BigQuery V2 Connector polls the status of the read job operation.</p> <p>Default is 10.</p> |

| Property | Description |
|---|---|
| Read Mode | <p>Specifies the read mode to read data from the Google BigQuery source.</p> <p>You can select one the following read modes:</p> <ul style="list-style-type: none"> - Direct. In direct mode, Google BigQuery V2 Connector reads data directly from the Google BigQuery source table. <p>Note: When you use hybrid and complex connection mode, you cannot use direct mode to read data from the Google BigQuery source.</p> <ul style="list-style-type: none"> - Staging. 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. <p>Default is Direct mode.</p> |
| Use EXPORT DATA Statement to stage | <p>Indicates whether to support ORDER BY clause in a custom query or SQL Override Query.</p> <p>This property applies to staging mode.</p> |
| Number of Threads for Downloading Staging Files | <p>Specifies the number of files that Google BigQuery V2 Connector downloads at a time to enable parallel download.</p> <p>This property applies to staging mode.</p> |
| Local Stage File Directory | <p>Specifies the directory on your local machine where Google BigQuery V2 Connector stores the Google BigQuery source data temporarily before it reads the data.</p> <p>This property applies to staging mode.</p> |
| Staging File Name | <p>Name of the staging file where data from the Google BigQuery source table is exported to Google Cloud Storage.</p> <p>This property applies to staging mode.</p> |
| Data Format of the staging file | <p>Specifies the data format of the staging file. You can select one of the following data formats:</p> <ul style="list-style-type: none"> - Avro - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - CSV. Supports flat data. <p>Note: In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ.</p> <ul style="list-style-type: none"> - Parquet <p>This property applies to staging mode.</p> |
| Enable Staging File Compression | <p>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.</p> <p>You can enable staging file compression to reduce cost and transfer time.</p> <p>This property applies to staging mode.</p> |
| Persist Extract Staging File After Download | <p>Indicates whether Google BigQuery V2 Connector must persist the staging file after it reads data from the staging file.</p> <p>By default, Google BigQuery V2 Connector deletes the staging file.</p> |
| Persist Destination Table | <p>Indicates whether Google BigQuery V2 Connector must persist the query results table after it reads data from the query results table.</p> <p>By default, Google BigQuery V2 Connector deletes the query results table.</p> |

| Property | Description |
|---------------------------------|---|
| pre SQL | <p>SQL statement that you want to run before reading data from the source.</p> <p>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:</p> <pre>SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;</pre> |
| pre SQL Configuration | <p>Specify a pre SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:PRESQL_SRC, DestinationDataset:EMPLOYEE, FlattenResults:False, WriteDisposition:WRITE_TRUNCATE, UseLegacySql:False</pre> |
| post SQL | <p>SQL statement that you want to run after reading data from the source.</p> <p>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:</p> <pre>UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL] SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where fullname='John Doe'</pre> |
| post SQL Configuration | <p>Specify a post SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:POSTSQL_SRC, DestinationDataset:EMPLOYEE, FlattenResults:True, WriteDisposition:WRITE_TRUNCATE, UseLegacySql:False</pre> |
| SQL Override Query | <p>Overrides the default SQL query used to read data from the Google BigQuery source.</p> <p>Note: When you specify SQL override query, you must specify a dataset name in the Source Dataset ID advanced source property.</p> <p>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.</p> <p>Ensure that you only map all the columns in the SQL override query to the target.</p> <p>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.</p> |
| Use Legacy SQL for SQL Override | <p>Indicates that the SQL Override query is specified in legacy SQL.</p> <p>Use the following format to specify a legacy SQL query for the SQL Override Query property:</p> <pre>SELECT <Col1, Col2, Col3> FROM [projectID:datasetID.tableName]</pre> <p>Clear this option to define a standard SQL override query.</p> <p>Use the following format to specify a standard SQL query for the SQL Override Query property:</p> <pre>SELECT * FROM `projectID.datasetID.tableName`</pre> |

| Property | Description |
|--------------------|--|
| Retry Options | <p>Comma-separated list to specify the following retry options:</p> <ul style="list-style-type: none"> - 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. <p>Use the following format to specify the retry options: For example,</p> <pre>RetryCount:5,RetryInterval:1,RetryExceptions:java.net.ConnectException java.io.IOException</pre> <p>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.</p> |
| Billing Project ID | <p>The project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs the query.</p> <p>If you do not specify a value for the Billing Project ID property, the Secure Agent runs the query in the Google Cloud project that contains the Google BigQuery objects based on the Project ID value specified in the Google BigQuery V2 connection.</p> |

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 | <p>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.</p> <p>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.</p> |

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

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: <ul style="list-style-type: none">- Return first row- Return last row- Return any row- Return all rows- Report error |

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 | 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 | 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. |

| Property | Description |
|---|--|
| Job Poll Interval in Seconds | 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: <ul style="list-style-type: none"> - 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. <ul style="list-style-type: none"> - Staging. 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 | 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 | 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 | 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 | Specifies the data format of the staging file. You can select one of the following data formats: <ul style="list-style-type: none"> - 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 | 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 | 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 | 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: <pre>SELECT * FROM [api-project-80697026669:EMPLOYEE.DEPARTMENT] LIMIT 1000;</pre> |

| Property | Description |
|---------------------------------|---|
| pre SQL Configuration | <p>Specify a pre SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:PRESQL_SRC, DestinationDataset:EMPLOYEE, FlattenResults:False, WriteDisposition:WRITE_TRUNCATE, UseLegacySql:False</pre> |
| post SQL | <p>SQL statement that you want to run after reading data from the source.</p> <p>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:</p> <pre>UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL] SET phoneNumber.number=1000011, phoneNumber.areaCode=100 where fullname='John Doe'</pre> |
| post SQL Configuration | <p>Specify a post SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:POSTSQL_SRC, DestinationDataset:EMPLOYEE, FlattenResults:True, WriteDisposition:WRITE_TRUNCATE, UseLegacySql:False</pre> |
| SQL Override Query | <p>Overrides the default SQL query used to read data from the Google BigQuery source.</p> <p>When you specify SQL override query, you must specify a dataset name in the Source Dataset ID advanced source property.</p> <p>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.</p> <p>Ensure that you only map all the columns in the SQL override query to the target.</p> |
| Use Legacy SQL for SQL Override | <p>Indicates that the SQL Override query is specified in legacy SQL.</p> <p>Use the following format to specify a legacy SQL query for the SQL Override Query property:</p> <pre>SELECT <Col1, Col2, Col3> FROM [projectID:datasetID.tableName]</pre> <p>Clear this option to define a standard SQL override query.</p> <p>Use the following format to specify a standard SQL query for the SQL Override Query property:</p> <pre>SELECT * FROM 'projectID.datasetID.tableName'</pre> |
| Retry Options | <p>Comma-separated list to specify the following retry options:</p> <ul style="list-style-type: none"> - 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. <p>Use the following format to specify the retry options:</p> <p>For example,</p> <pre>RetryCount:5, RetryInterval:1, RetryExceptions:java.net.ConnectException java.io.IOException</pre> <p>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.</p> |

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.

The screenshot shows the Looker Studio interface. On the left is a vertical toolbar with icons for Source, Target, Aggregator, Cleanse, Data Masking, Expression, and Filter. The main workspace contains a data pipeline diagram with three components: 'Source', 'Expression' (containing the formula $f(x)$), and 'Target', connected by arrows. Below the pipeline is a separate box labeled 'U_Lkp_GBQv2'. At the bottom, the 'Properties' panel is open for the 'U_Lkp_GBQv2' transformation. The 'General' tab is selected, showing fields for 'Name' (U_Lkp_GBQv2), 'Description', and a checked '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.

This screenshot shows the 'Incoming Fields' tab of the 'U_Lkp_GBQv2' transformation in the Properties panel. The 'Incoming Field' dropdown is set to 'Not Parameterized'. Below, the 'Incoming Fields' section has a table with one row: 'SRC_slna'.

| Field Name |
|------------|
| SRC_slna |

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

Properties | U_Lkp_GBQv2

General

Incoming Fields

Lookup Object

Lookup Condition

Return Fields

Advanced

Lookup Object Details

Connection: sd_gbq_v2_Simple (Google Big Query ... View... New Connection... New Parameter...

Source Type: Single Object

Lookup Object: sd_gbq_ds/sd_multi Select... Preview Data...

Multiple Matches: Return any row

Advanced

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

- 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.

Properties | U_Lkp_GBQv2

General

Incoming Fields

Lookup Object

Lookup Condition

Return Fields

Advanced

Lookup Condition: Simple

Lookup Conditions

| Lookup Field | Operator | Incoming Field |
|--------------|----------|----------------|
| slns | = | SRC_slns |

- 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.

Field Expression: Result_Final(string, 50, 0)

Configure expression by adding fields and functions.

Expression: Not Parameterized

Fields

- SRC_slns
- SRC_name
- SRC_m1
- SRC_m2

Expression

```

iif(
  (SRC_m1+SRC_m2)>170,
  :LKP.U_Lkp_GBQv2(SRC_slns) || 'is passed with this marks',
  'Failed'
)

```

Validate

OK Cancel

- 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.

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.

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. |

| Property | Description |
|-----------------------|--|
| Parameter | <p>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.</p> <p>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.</p> <p>Does not apply when you perform a data driven operation.</p> |
| Object | Name of the Google BigQuery target object based on the target type selected. |
| Create New at Runtime | <p>Creates a target.</p> <p>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.</p> <p>You must specify a valid dataset ID for the Path attribute.</p> <p>The target name can contain alphanumeric characters. You cannot use special characters in the file name except the underscore character (_).</p> <p>You cannot parameterize the target at runtime.</p> |
| Operation | <p>You can select one the following operations:</p> <ul style="list-style-type: none"> - Insert - Update - Upsert (Update or Insert) - Delete - Data Driven <p>Note: If you use complex connection mode, you cannot configure update, upsert, and delete operations.</p> |
| Data Driven Condition | <p>Flags rows for an insert, update, delete, or reject operation based on the data driven expression you specify.</p> <p>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.</p> <p>Note: Appears only when you select Data Driven as the operation type.</p> |
| Update Columns | <p>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.</p> <p>You can select multiple columns. By default, no columns are specified.</p> |

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 | <p>Determines the mode that the Secure Agent uses to update rows in the Google BigQuery target.</p> <p>You can select one of the following modes:</p> <ul style="list-style-type: none"> - 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. <p>Default is Update as Update.</p> <p>Not applicable when you perform a data driven operation.</p> |

| Property | Description |
|---------------------------|---|
| Enable Data Driven | 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. |
| 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 not selected. |
| 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 | Optional. Overrides the update SQL statement that the Secure Agent generates to update the Google BigQuery target. Use the following format to define an update override query: <pre>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></pre> For example, <pre>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</pre> 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 | 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: <ul style="list-style-type: none"> - 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 | <p>Specifies how Google BigQuery V2 Connector must write data in bulk mode if the target table already exists.</p> <p>You can select one of the following values:</p> <ul style="list-style-type: none"> - 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. <p>Write disposition is applicable for bulk mode.</p> <p>Write disposition is applicable only when you perform an insert operation on a Google BigQuery target.</p> |
| Write Mode | <p>Specifies the mode to write data to the Google BigQuery target.</p> <p>You can select one of the following modes:</p> <ul style="list-style-type: none"> - 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. - Streaming¹. Google BigQuery V2 Connector directly writes data to the BigQuery target. Google BigQuery V2 Connector writes the data into the target row by row. - CDC¹. 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. <p>Default is Bulk mode.</p> <p>Streaming mode is not applicable when you perform a data driven operation.</p> |
| Streaming Template Table Suffix | <p>Specify the suffix to add to the individual target tables that Google BigQuery V2 Connector creates based on the template target table.</p> <p>This property applies to streaming mode.</p> <p>If you select the Enable Merge option, Google BigQuery V2 Connector ignores this property.</p> <p>Streaming mode is not applicable when you perform a data driven operation.</p> |
| Rows per Streaming Request | <p>Specifies the number of rows that Google BigQuery V2 Connector streams to the BigQuery target for each request.</p> <p>Default is 500 rows.</p> <p>The maximum row size that Google BigQuery V2 Connector can stream to the Google BigQuery target for each request is 10 MB.</p> <p>This property applies to streaming mode.</p> <p>Streaming mode is not applicable when you perform a data driven operation.</p> |
| Staging file name | <p>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.</p> <p>This property applies to bulk mode.</p> |

| Property | Description |
|--|---|
| Data Format of the staging file | <p>Specifies the data format of the staging file. You can select one of the following data formats:</p> <ul style="list-style-type: none"> - Avro - JSON (Newline Delimited). Supports flat and record data with nested and repeated fields. - Parquet - CSV. Supports flat data. <p>Note: In a .csv file, columns of the Timestamp data type are represented as floating point numbers that cause the milliseconds value to differ.</p> <p>This property applies to bulk and CDC mode.</p> <p>Avro and parquet format is not applicable when you perform a data driven operation.</p> |
| Persist Staging File After Loading | <p>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.</p> <p>By default, Google BigQuery V2 Connector deletes the staging file in Google Cloud Storage.</p> <p>This property applies to bulk mode.</p> |
| Enable Staging File Compression | <p>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.</p> <p>You can enable staging file compression to reduce cost and transfer time.</p> |
| Job Poll Interval in Seconds | <p>The number of seconds after which Google BigQuery V2 Connector polls the status of the write job operation.</p> <p>Default is 10.</p> |
| Number of Threads for Uploading Staging file | <p>The number of files that Google BigQuery V2 Connector must create to upload the staging file in bulk mode.</p> |
| Local Stage File Directory | <p>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.</p> <p>This property applies to bulk mode.</p> |
| Allow Quoted Newlines | <p>Indicates whether Google BigQuery V2 Connector must allow the quoted data sections with newline character in a .csv file.</p> |
| Field Delimiter | <p>Indicates whether Google BigQuery V2 Connector must allow field separators for the fields in a .csv file.</p> |
| Quote Character | <p>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.</p> <p>Default is double quotes.</p> |
| Allow Jagged Rows | <p>Indicates whether Google BigQuery V2 Connector must accept the rows without trailing columns in a .csv file.</p> |
| Pre SQL | <p>SQL statement that you want to run before writing data to the target.</p> <p>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:</p> <pre>SELECT * FROM 'api-project-80697026669.EMPLOYEE.RegionNation' LIMIT 1000</pre> |

| Property | Description |
|--------------------------------------|--|
| Pre SQL Configuration | <p>Specify a pre-SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:PRESQL_TGT2, DestinationDataset:EMPLOYEE, FlattenResults:False, WriteDisposition:WRITE_TRUNCATE, UseLegacySql:False</pre> |
| Post SQL | <p>SQL statement that you want to run after writing the data into the target.</p> <p>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:</p> <pre>UPDATE [api-project-80697026669.EMPLOYEE.PERSONS_TGT_DEL] SET phoneNumber.number =1000011, phoneNumber.areaCode=100 where fullname='John Doe'</pre> |
| Suppress post SQL on Error | <p>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.</p> <p>Default is not selected.</p> |
| Post SQL Configuration | <p>Specify a post-SQL configuration.</p> <p>For example,</p> <pre>DestinationTable:POSTSQL_SRC, DestinationDataset:EMPLOYEE, FlattenResults:True, UseLegacySQL:False</pre> |
| Truncate target table | <p>Truncates the Google BigQuery target table before loading data to the target.</p> <p>Default is not selected.</p> |
| Allow Duplicate Inserts ¹ | <p>Applicable when the selected data format for the staging file is CSV when the mapping contains both connected and unconnected ports.</p> <p>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.</p> <p>When you do not enable the Use Default Column Values option, the agent populates the target with values from the connected ports.</p> <p>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.</p> |
| Disable Duplicate Update Rows | <p>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.</p> <p>Select this option to configure the mapping to process only one of the incoming rows and ignore the rest of the incoming rows.</p> <p>Default is not selected.</p> |
| Forward Rejected Rows | <p>Applicable only when you configure DD_REJECT constant in the data driven condition to reject all the rows.</p> <p>Otherwise, this property is not applicable for Google BigQuery V2 Connector.</p> |
| Billing Project ID | <p>The project ID for the Google Cloud project that is linked to an active Google Cloud Billing account where the Secure Agent runs the query.</p> <p>If you do not specify a value for the Billing Project ID property, the Secure Agent runs the query in the Google Cloud project that contains the Google BigQuery objects based on the Project ID value specified in the Google BigQuery V2 connection.</p> |

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.
- 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`.



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 |

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.

PropertiesPreviewSQL

GeneralIncoming FieldsSQLInput FieldsField MappingOutput FieldsAdvanced

Connection:Auto_GBQv2_HybridView...New Connection...New Parameter...

SQL Type:Stored Procedure

Stored Procedure:StoredProc_Dataset/IntegerProcedureSelect...

Description:IntegerProcedure

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 `SQLException` output field.

The screenshot shows the configuration window for an SQL transformation. The 'SQL' tab is selected, displaying a 'Query' section with a text area containing the following SQL query:

```
1 select
2   Fld_String
3 from
4   Datatypes.MasterData
5 where
6   Fld_string is not null
7   and Fld_String != ''
```

Buttons for 'Format SQL' and 'Validate' are located to the right of the query text area. The left sidebar contains tabs for 'General', 'Incoming Fields', 'Field Mapping', 'Output Fields', and 'Advanced'. The 'Field Mapping' section is expanded, showing a list of fields: Fld_String, Fld_Integer, Fld_Float, Fld_Boolean, Fld_Timestamp, Fld_Date, Fld_Time, and Fld_DateTime.

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.

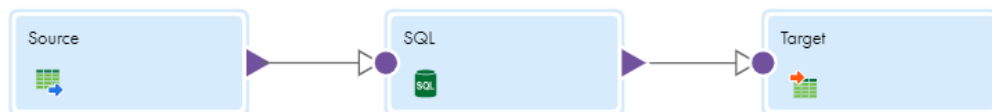
Configuring an SQL transformation

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

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, you must configure the following properties in the Google BigQuery V2 connection:

| Connection Property | Value |
|-----------------------------|----------------------------------|
| Optional Properties | Required |
| 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> |

| Mapping Configuration | Staging Object Type | Staging Object Name Format |
|--|---|--|
| 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> |
| Pushdown optimization with Google Cloud Storage source and Google BigQuery target | Staging table | <Informatica_Prefix>_<TargetTableName>_PDO_<TimestampUptoMilliseconds>_<UUID> |
| Pushdown optimization with Google BigQuery source and target | Staging table | <Informatica_Prefix>_<TargetTableName>_PDO_<TimestampUptoMilliseconds>_<UUID> |
| Pushdown 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>_<PartitionID^>_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

Rules and Guidelines for mapping and mapping tasks

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

Sources and targets

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

- 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 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 the 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 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 datatype for SESSSTARTTIME is string, you might see difference in data format when you compare a mapping that runs with full pushdown optimization and a mapping that runs without pushdown optimization. For example, SESSSTARTTIME returns DateTime value in the MM/DD/YYYY HH24:MI:SS format when a mapping runs with full pushdown optimization. When you run the same mapping without pushdown optimization, SESSSTARTTIME appends additional zeroes to the return value, MM/DD/YYYY HH24:MI:SS.000000.
- When you run a mapping without pushdown optimization, you must use only the columns which are mapped in the Update Column field.

- You must select **is expression variable** in-out parameter property to read a parameter value as an expression in a Filter transformation in non-pushdown optimization mode.

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 DateTime 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 a source column does not contain a NULL value in the first row. Otherwise, the mapping fails.
 If the source contains a NULL value in any other rows of the column, the mapping considers the filter condition for the previous row. This results in data duplication in the lookup table.
- 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.informatika.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 configuring 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.
```

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 using 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.

APPENDIX A

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 | 1 to 104,857,600 characters Default precision is 255. You can increase the value up to 104857600 characters. |
| 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 |

| Google BigQuery Data Type | Transformation Data Type | Range and Description for the Transformation Data Type |
|--|--------------------------|--|
| BOOL ² | String | Boolean True or False values. Default precision is 5. |
| ¹ where <ul style="list-style-type: none"> - 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 upto six fractional digits. - [time zone] represents the time zone. Default time zone is UTC. ² Applies only to SQL transformation. | | |

INDEX

B

bulk mode
staging file [45](#)

C

cache
enable lookup cache [63](#)
CDC source
Google BigQuery V2 mapping task [69](#)
Cloud Application Integration community
URL [5](#)
Cloud Developer community
URL [5](#)
connections
Google BigQuery [14](#)
custom query [42](#)

D

data filters [47](#)
Data Integration community
URL [5](#)
dynamic schema handling [51](#)

G

Google BigQuery
connection properties [14](#)
pushdown through Redshift V2 Connection [20](#)
Google BigQuery connection
pushdown optimization overview [18](#)
Google BigQuery connector
administration [8](#)
Google BigQuery data types
overview [83](#)
Google BigQuery V2 connection
configuration [22](#)
pushdown optimization overview [18](#)
Google BigQuery V2 connections
overview [9](#)
Google BigQuery V2 connector
overview [7](#)
supported task and object types [7](#)
Google BigQuery V2 data types
mapping to transformation data types [83](#)
Google BigQuery V2 lookups
mapping tasks [60](#)

I

Informatica Global Customer Support
contact information [6](#)
Informatica Intelligent Cloud Services
web site [5](#)

L

lookup
multiple matches [56](#)
lookup caches
dynamic [63](#)
persistent [63](#)
Lookup transformation
lookup caching [63](#)

M

maintenance outages [6](#)
mapping
Google BigQuery sources [52](#)
Google BigQuery targets [63](#)
mappings
lookup overview [56](#)
lookup properties [56](#)

P

post-SQL commands
entering [47](#)
pre-SQL and post-SQL
entering [47](#)
pre-SQL commands
entering [47](#)
pushdown optimization
functions [22, 24, 26](#)
transformations [22, 24, 26](#)
Pushdown optimization
preview [20](#)
Pushdown Optimization
Rules and Guidelines for Functions [34](#)
Rules and Guidelines for mappings with Amazon S3 [31](#)
Pushdown optimization preview [20](#)

S

SQL transformations
configuration [73](#)
selecting a stored procedure [74](#)
status
Informatica Intelligent Cloud Services [6](#)

streaming mode
 creating template table [45](#)
system status [6](#)

T

transformations
 pushdown optimization [24](#)
troubleshooting
 mapping task [82](#)
trust site
 description [6](#)

U

upgrade notifications [6](#)
upsert [71](#)

W

web site [5](#)
write modes
 bulk [45](#)
 streaming [45](#)