Magnitude Simba Google BigQuery ODBC Data Connector

Installation and Configuration Guide

Version 2.3.3

February 26, 2021

Contact Us

Magnitude Software, Inc.

www.magnitude.com

About This Guide

Purpose

The *Magnitude Simba Google BigQuery ODBC Data Connector Installation and Configuration Guide* explains how to install and configure the Magnitude Simba Google BigQuery ODBC Data Connector. The guide also provides details related to features of the connector.

Audience

The guide is intended for end users of the Simba Google BigQuery ODBC Connector, as well as administrators and developers integrating the connector.

Knowledge Prerequisites

To use the Simba Google BigQuery ODBC Connector, the following knowledge is helpful:

- Familiarity with the platform on which you are using the Simba Google BigQuery ODBC Connector
- Ability to use the data source to which the Simba Google BigQuery ODBC Connector is connecting
- An understanding of the role of ODBC technologies and driver managers in connecting to a data source
- Experience creating and configuring ODBC connections
- Exposure to SQL

Document Conventions

Italics are used when referring to book and document titles.

Bold is used in procedures for graphical user interface elements that a user clicks and text that a user types.

`Monospace font` indicates commands, source code, or contents of text files.

> **Note:**
>
> A text box with a pencil icon indicates a short note appended to a paragraph.

! Important:

A text box with an exclamation mark indicates an important comment related to the preceding paragraph.

Table of Contents

About the Simba Google BigQuery ODBC Connector

The Simba Google BigQuery ODBC Connector enables Business Intelligence (BI), analytics, and reporting on data that has been uploaded to Google Storage. The connector complies with the ODBC 3.80 data standard and adds important functionality such as Unicode, as well as 32- and 64-bit support for high-performance computing environments on all platforms.

ODBC is one of the most established and widely supported APIs for connecting to and working with databases. At the heart of the technology is the ODBC connector, which connects an application to the database. For more information about ODBC, see *Data Access Standards* on the Simba Technologies website: https://www.simba.com/resources/data-access-standards-glossary. For complete information about the ODBC specification, see the *ODBC API Reference* from the Microsoft documentation: https://docs.microsoft.com/en-us/sql/odbc/reference/syntax/odbc-api-reference.

The Simba Google BigQuery ODBC Connector is available for Microsoft® Windows®, Linux, and macOS platforms.

The *Installation and Configuration Guide* is suitable for users who are looking to access BigQuery data from their desktop environment. Application developers might also find the information helpful. Refer to your application for details on connecting via ODBC.

Note:

For information about how to use the connector in various BI tools, see the *Simba ODBC Connectors Quick Start Guide for Windows*: http://cdn.simba.com/docs/ODBC_QuickstartGuide/content/quick_start/intro.htm.

Windows Connector

Windows System Requirements

Install the connector on client machines where the application is installed. Before installing the connector, make sure that you have the following:

- Administrator rights on your machine.
- A machine that meets the following system requirements:
  - One of the following operating systems:
    - Windows 10, 8.1, or 7 SP1
    - Windows Server 2016, 2012, or 2008 R2 SP1
  - 100 MB of available disk space

Before the connector can be used, the Visual C++ Redistributable for Visual Studio 2015 with the same bitness as the connector must also be installed. If you obtained the connector from the Simba website, then your installation of the connector automatically includes this dependency. Otherwise, you must install the redistributable manually. You can download the installation packages for the redistributable at https://www.microsoft.com/en-ca/download/details.aspx?id=48145.

Installing the Connector on Windows

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

On 64-bit Windows operating systems, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectorss. Make sure that you use a connector whose bitness matches the bitness of the client application:

- `SimbaODBCDriverforGoogleBigQuery32.msi` for 32-bit applications
- `SimbaODBCDriverforGoogleBigQuery64.msi` for 64-bit applications

You can install both versions of the connector on the same machine.

To install the Simba Google BigQuery ODBC Connector on Windows:

1. Depending on the bitness of your client application, double-click to run **SimbaODBCDriverforGoogleBigQuery32.msi** or **SimbaODBCDriverforGoogleBigQuery64.msi**.

2. Click **Next**.
3. Select the check box to accept the terms of the License Agreement if you agree, and then click **Next**.
4. To change the installation location, click **Change**, then browse to the desired folder, and then click **OK**. To accept the installation location, click **Next**.
5. Click **Install**.
6. When the installation completes, click **Finish**.
7. If you received a license file through email, then copy the license file into the `\lib` subfolder of the installation folder you selected above. You must have Administrator privileges when changing the contents of this folder.

Creating a Data Source Name on Windows

Typically, after installing the Simba Google BigQuery ODBC Connector, you need to create a Data Source Name (DSN).

Alternatively, for information about DSN-less connections, see Using a Connection String on page 45.

To create a Data Source Name on Windows:

1. From the Start menu, go to **ODBC Data Sources**.

   Note:

   Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to BigQuery.

2. In the ODBC Data Source Administrator, click the **Drivers** tab, and then scroll down as needed to confirm that the Simba Google BigQuery ODBC Connector appears in the alphabetical list of ODBC connectors that are installed on your system.
3. Choose one:
   - To create a DSN that only the user currently logged into Windows can use, click the **User DSN** tab.
   - Or, to create a DSN that all users who log into Windows can use, click the **System DSN** tab.

   Note:

   It is recommended that you create a System DSN instead of a User DSN. Some applications load the data using a different user account, and might not be able to detect User DSNs that are created under another user account.

4. Click **Add**.
5. In the Create New Data Source dialog box, select **Simba Google BigQuery ODBC Connector** and then click **Finish**. The Simba Google BigQuery ODBC Connector DSN Setup dialog box opens.
6. In the **Data Source Name** field, type a name for your DSN.
7. Optionally, in the **Description** field, type relevant details about the DSN.
8. Configure authentication using the options in the Authentication area. For more information, see Configuring Authentication on Windows on page 11.
9. To allow the connector to access Google Drive so that it can support federated tables that combine BigQuery data with data from Google Drive, select the **Request Google Drive Scope Access** check box.
10. Choose one:
    - To verify the server using the trusted CA certificates from a specific `.pem` file, specify the full path to the file in the **Trusted Certificates** field and leave the **Use System Trust Store** check box cleared.
    - Or, to use the trusted CA certificates `.pem` file that is installed with the connector, leave the default value in the **Trusted Certificates** field, and leave the **Use System Trust Store** check box cleared.
    - Or, to use the Windows Trust Store, select the **Use System Trust Store** check box and leave the **Trusted Certificates** field cleared.
11. From the **Minimum TLS** drop-down list, select the minimum version of TLS to use when connecting to your data store.
12. In the **Catalog (Project)** drop-down list, select the name of your BigQuery project. This project is the default project that the Simba Google BigQuery ODBC Connector queries against, and also the project that is billed for queries that are run using the DSN.

    > **Note:**
    >
    > If you are not signed in to your Google account, then you are prompted to sign in.

13. Optionally, in the **Dataset** drop-down list, select the name of the dataset the connector will query by default. For more information, see Dataset on page 63.
14. To configure a connection through a proxy server, click **Proxy Options**. For more information, see Configuring a Proxy Connection on Windows on page 14.
15. To configure logging behavior for the connector, click **Logging Options**. For more information, see Configuring Logging Options on Windows on page 18.
16. To configure advanced connector options, click **Advanced Options**. For more information, see Configuring Advanced Options on Windows on page 14.
17. To test the connection, click **Test**. Review the results as needed, and then click **OK**.

18. To save your settings and close the Simba Google BigQuery ODBC Connector DSN Setup dialog box, click **OK**.
19. To close the ODBC Data Source Administrator, click **OK**.

Configuring Authentication on Windows

The Simba Google BigQuery ODBC Connector uses the OAuth 2.0 protocol for authentication and authorization. It authenticates your connection through Google OAuth APIs. You can configure the connector to provide your credentials and authenticate the connection to the database using one of the following methods:

- Using a Google User Account on page 11
- Using a Google Service Account on page 13

Using a Google User Account

You can configure the connector to authenticate the connection with a Google user account. This authentication method uses the OAuth 2.0 access and refresh tokens associated with the user account as the credentials.

The access token is transmitted with every API call that the connector makes, and it is required for accessing BigQuery data stores. However, the access token expires after a certain amount of time and must be renewed using the refresh token. If the refresh token is stored in your connection information, the connector automatically uses it to renew access tokens when they expire.

Note:

For more information about OAuth 2.0, see "Using OAuth 2.0 to Access Google APIs" in the Google Identity Platform documentation: https://developers.google.com/identity/protocols/OAuth2.

At minimum, you need to provide the OAuth 2.0 refresh token associated with your account. The connector retrieves and uses an access token based on your specified refresh token.

- If you do not have your refresh token, see Retrieving a Refresh Token on page 12.
- If you already have your refresh token, see Providing a Refresh Token on page 12.
- If you want to provide a `.json` key file that contains your credentials instead of providing your refresh token directly in your connection information, see Providing a Key File on page 13.

Retrieving a Refresh Token

When you authenticate your connection this way, the authentication process provides a temporary confirmation code that you can exchange for an access token and a refresh token.

To configure user account authentication by retrieving a refresh token on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
2. From the **OAuth Mechanism** drop-down list, select **User Authentication**.
3. Click **Sign In**.
4. In the browser that opens, type your credentials for accessing your BigQuery data and sign in to your account.
5. When you are prompted to allow BigQuery Client Tools to access your data in Google BigQuery, click **Accept**.

   The browser displays a confirmation code.
6. Copy and paste the code into the **Confirmation Code** field in the Simba Google BigQuery ODBC Connector DSN Setup dialog box.
7. Click inside the **Refresh Token** field or press **TAB** to move your caret from the Confirmation Code field into the Refresh Token field.

   The connector automatically populates the field with your refresh token. The refresh token is used whenever the connector needs to access your BigQuery data. You can save the refresh token in the DSN so that you only need to generate it once.

   > **Note:**
   >
   > A confirmation code can only be used once. You must get a new confirmation code from Google whenever you need another refresh token.

Providing a Refresh Token

If you already have your refresh token, then you can provide the token in your connection information without going through the retrieval process described above.

To configure user account authentication by providing a refresh token on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
2. From the **OAuth Mechanism** drop-down list, select **User Authentication**.
3. In the **Refresh Token** field, type the refresh token associated with your user account.

Providing a Key File

As an alternative to providing your refresh token directly in your connection information, you can save the token in a `.json` key file and then specify the path to the file in your connection information.

The file must define a JSON object of type `authorized_user` containing the refresh token, client ID, and client secret associated with your user account. For example, the `.json` key file must be written in the following format:

```
{
   "type": "authorized_user",
   "client_id": "[YourClientID]",
   "client_secret":"[YourClientSecret]",
   "refresh_token":"[YourRefreshToken]"
}
```

To configure user account authentication by providing a key file on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
2. From the **OAuth Mechanism** drop-down list, select **Service Authentication**.

   > **Note:**
   >
   > Although this is a form of user authentication, the key file must be provided using the service authentication options.

3. In the **Email** field, type your user account email ID.
4. In the **Key File Path** field, type the full path to the `.json` key file.

Using a Google Service Account

You can configure the connector to authenticate the connection with a Google service account. When you authenticate your connection this way, the connector handles authentication on behalf of the service account, so that an individual user account is not directly involved and no user input is required.

To authenticate your connection this way, you must provide a Google service account email address and the full path to a private key file for the service account. You can generate and download the private key file when you set up the service account.

Note:

- For more information about OAuth 2.0 authentication using a service account, see "Using OAuth 2.0 for Server to Server Applications" in the Google Identity Platform documentation: https://developers.google.com/identity/protocols/OAuth2ServiceAccount.
- For information about obtaining service account keys, see "Creating and Managing Service Account Keys" in the Google Cloud Identity & Access Management documentation: https://cloud.google.com/iam/docs/creating-managing-service-account-keys.

To configure service account authentication on Windows:

1. To access authentication options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, and then click **Configure**.
2. From the **OAuth Mechanism** drop-down list, select **Service Authentication**.
3. In the **Email** field, type your service account email ID.
4. In the **Key File Path** field, type the full path to the `.p12` or `.json` key file that is used to authenticate the service account ID.

Configuring a Proxy Connection on Windows

If you are connecting to the data source through a proxy server, you must provide connection information for the proxy server.

To configure a proxy server connection on Windows:

1. To access proxy server options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Proxy Options**.
2. Select the **Use Proxy Server** check box.
3. In the **Proxy Host** field, type the host name or IP address of the proxy server.
4. In the **Proxy Port** field, type the number of the TCP port that the proxy server uses to listen for client connections.
5. In the **Proxy Username** field, type your user name for accessing the proxy server.
6. In the **Proxy Password** field, type the password corresponding to the user name.
7. To save your settings and close the Proxy Options dialog box, click **OK**.

Configuring Advanced Options on Windows

You can configure advanced options to modify the behavior of the connector.

To configure advanced options on Windows:

1. To access advanced options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Advanced Options**.
2. From the **Language Dialect** drop-down list, select the SQL syntax to use when executing queries:
   - To use standard SQL syntax, select **Standard SQL**.
   - Or, to use the legacy BigQuery SQL syntax, select **Legacy SQL**.
3. To allow query results that are larger than 128MB in size when using Legacy SQL, select the **Allow Large Result Sets** check box.

   > **Note:**
   >
   > This option is only available if the Language Dialect drop-down list is set to Legacy SQL. Large result sets are always enabled if the Language Dialect drop-down list is set to Standard SQL.

4. To specify the dataset that stores temporary tables for large result sets, do one of the following:
   - To use the default dataset with the ID _bqodbc_temp_tables, select the **Use Default _bqodbc_temp_tables Large Results Dataset** check box.
   - Or, to specify a different dataset, clear the **Use Default _bqodbc_temp_tables Large Results Dataset** check box and, in the **Dataset Name For Large Result Sets** field, type the ID of the BigQuery dataset that you want to use.

   > **Note:**
   >
   > - If the dataset does not exist and the data store specifies the US region, the connector creates the dataset.
   > - These options are only available if the connector is configured to use large result sets.

5. In the **Temporary Table Expiration Time** field, type the length of time (in milliseconds) for which any temporary tables exist. The minimum value is 3600000.

   > **Note:**
   >
   > This option is only available if the connector is configured to use large result sets.

6. In the **Rows Per Block** field, type the maximum number of rows to fetch for each data request.

7. To use the High-Throughput API to handle large result sets more efficiently, select the **Enable High-Throughput API** check box. For more information, including prerequisites and detailed instructions, see Configuring the High-Throughput API on Windows on page 16.
8. To use a customer-managed encryption key (CMEK) when executing queries, in the **Path To CMEK** field, type the resource ID of the CMEK. For more information, see "Protecting Data with Cloud KMS Keys" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/customer-managed-encryption.

   Important:
   - Do not specify a CMEK unless you are certain that it is the correct value to use. If you execute an INSERT statement with an incorrect CMEK, the connector returns an error or corrupts the table.
   - The connector uses the specified CMEK for all queries.
9. In the **Default String Column Length** field, type the maximum number of characters that can be contained in STRING columns.
10. To return data as SQL_WVARCHAR data instead of SQL_VARCHAR data, select the **Use SQL_WVARCHAR Instead Of SQL_VARCHAR** check box.

    Note:

    This option applies only to result set columns that the connector would normally return as SQL_VARCHAR columns. It does not convert all columns into SQL_WVARCHAR.
11. To access public projects and use them as catalogs for the connection, in the **Additional Projects** field, type a comma-separated list of project names.
12. To pass properties down to the server when inserting a job, in the **Query Properties** field, type a comma-separated list of key value pairs.
13. To save your settings and close the Advanced Options dialog box, click **OK**.

Configuring the High-Throughput API on Windows

You can configure the connector to use the High-Throughput API to handle large result sets more efficiently. For more information about the High-Throughput API, see High-Throughput API on page 55.

To configure the High-Throughput API on Windows:

1. Make sure that your Google BigQuery project has the Storage API enabled. For more information about the Storage API, see "BigQuery Storage API Overview" in the Google BigQuery documentation:

https://cloud.google.com/bigquery/docs/reference/storage/.

2. To access the DSN that you want to configure the High-Throughput API for, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**.
3. Choose one:
   - To verify the server using the trusted CA certificates from a specific `.pem` file, specify the full path to the file in the **Trusted Certificates** field and leave the **Use System Trust Store** check box cleared.
   - Or, to use the trusted CA certificates `.pem` file that is installed with the connector, leave the default value in the **Trusted Certificates** field, and leave the **Use System Trust Store** check box cleared.

   Important:

   Do not select the Use System Trust Store check box. The High-Throughput API is not compatible with the Windows Trust Store.
4. Click **Advanced Options**.
5. If you are using Legacy SQL (the **Language Dialect** drop-down list is set to **Legacy SQL**), then make sure that the **Allow Large Result Sets** check box is selected.
6. To specify the dataset that stores temporary tables for large result sets and result sets returned by the High-Throughput API, do one of the following:
   - To use the default dataset with the ID _bqodbc_temp_tables, select the **Use Default _bqodbc_temp_tables Large Results Dataset** check box.
   - Or, to specify a different dataset, clear the **Use Default _bqodbc_temp_tables Large Results Dataset** check box and, in the **Dataset Name For Large Result Sets** field, type the ID of the BigQuery dataset that you want to use.

   Note:
   - If the dataset does not exist and the data store specifies the US region, the connector creates the dataset.
   - These options are only available if the connector is configured to use large result sets.
7. Select the **Enable High-Throughput API** check box.
8. In the **Minimum Query Results Size for HTAPI** field, specify the minimum number of table rows required to activate the High-Throughput API.
9. In the **Ratio of Results to Rows Per Block** field, specify the minimum ratio of total rows to rows in the first page required to activate reading through the High-Throughput API.

> **Note:**
>
> If this value is set to 0, then the connector uses the High-Throughput API for all query results that meet the minimum results size specified by Minimum Query Results Size for HTAPI.

10. To save your settings and close the Advanced Options dialog box, click **OK**.

The connector now uses the BigQuery High-Throughput API instead of the REST API for requests where both:

- the number of table rows in your query results exceeds the Minimum Query Results Size for HTAPI number;
- and the number of pages in the results exceeds the Ratio of Results to Rows Per Block value.

Configuring Logging Options on Windows

To help troubleshoot issues, you can enable logging. In addition to functionality provided in the Simba Google BigQuery ODBC Connector, the ODBC Data Source Administrator provides tracing functionality.

> **Important:**
>
> Only enable logging or tracing long enough to capture an issue. Logging or tracing decreases performance and can consume a large quantity of disk space.

Configuring Connector-wide Logging Options

The settings for logging apply to every connection that uses the Simba Google BigQuery ODBC Connector, so make sure to disable the feature after you are done using it. To configure logging for the current connection, see Configuring Logging for the Current Connection on page 20.

To enable connector-wide logging on Windows:

1. To access logging options, open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select the logging level corresponding to the amount of information that you want to include in log files:

Logging Level	Description
OFF	Disables all logging.
FATAL	Logs severe error events that lead the connector to abort.
ERROR	Logs error events that might allow the connector to continue running.
WARNING	Logs events that might result in an error if action is not taken.
INFO	Logs general information that describes the progress of the connector.
DEBUG	Logs detailed information that is useful for debugging the connector.
TRACE	Logs all connector activity.

3. In the **Log Path** field, specify the full path to the folder where you want to save log files. You can type the path into the field, or click **Browse** and then browse to select the folder.
4. In the **Max Number Files** field, type the maximum number of log files to keep.

   Note:

   After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

5. In the **Max File Size** field, type the maximum size of each log file in megabytes (MB).

   Note:

   After the maximum file size is reached, the connector creates a new file and continues logging.

6. Click **OK**.
7. Restart your ODBC application to make sure that the new settings take effect.

The Simba Google BigQuery ODBC Connector produces the following log files at the location you specify in the Log Path field:

- A `simbabigqueryodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `simbabigqueryodbcdriver_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

To disable connector logging on Windows:

1. Open the ODBC Data Source Administrator where you created the DSN, then select the DSN, then click **Configure**, and then click **Logging Options**.
2. From the **Log Level** drop-down list, select **LOG_OFF**.
3. Click **OK**.
4. Restart your ODBC application to make sure that the new settings take effect.

Configuring Logging for the Current Connection

You can configure logging for the current connection by setting the logging configuration properties in the DSN or in a connection string. For information about the logging configuration properties, see Configuring Logging Options on Windows on page 18. Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

> **Note:**
>
> If the LogLevel configuration property is passed in via the connection string or DSN, the rest of the logging configurations are read from the connection string or DSN and not from the existing connector-wide logging configuration.

To configure logging properties in the DSN, you must modify the Windows registry. For information about the Windows registry, see the Microsoft Windows documentation.

> **Important:**
>
> Editing the Windows Registry incorrectly can potentially cause serious, system-wide problems that may require re-installing Windows to correct.

To add logging configurations to a DSN on Windows:

1. On the Start screen, type **regedit**, and then click the **regedit** search result.
2. Navigate to the appropriate registry key for the bitness of your connector and your machine:
   - 32-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI\[DSN Name]**
   - 64-bit System DSNs: **HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\[DSN Name]**

- 32-bit and 64-bit User DSNs: **HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI\[DSN Name]**

3. For each configuration option that you want to configure for the current connection, create a value by doing the following:
   a. If the key name value does not already exist, create it. Right-click the *[DSN Name]* and then select **New > String Value**, type the key name of the configuration option, and then press **Enter**.
   b. Right-click the key name and then click **Modify**.

      To confirm the key names for each configuration option, see Connector Configuration Options on page 59.
   c. In the Edit String dialog box, in the **Value Data** field, type the value for the configuration option.
4. Close the Registry Editor.
5. Restart your ODBC application to make sure that the new settings take effect.

Verifying the Connector Version Number on Windows

If you need to verify the version of the Simba Google BigQuery ODBC Connector that is installed on your Windows machine, you can find the version number in the ODBC Data Source Administrator.

To verify the connector version number on Windows:

1. From the Start menu, go to **ODBC Data Sources**.

   > **Note:**
   >
   > Make sure to select the ODBC Data Source Administrator that has the same bitness as the client application that you are using to connect to BigQuery.

2. Click the **Drivers** tab and then find the Simba Google BigQuery ODBC Connector in the list of ODBC connectors that are installed on your system. The version number is displayed in the **Version** column.

macOS Connector

macOS System Requirements

Install the connector on client machines where the application is installed. Each client machine that you install the connector on must meet the following minimum system requirements:

- macOS version 10.13 or 10.14 or 10.15
- 150MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later

Installing the Connector on macOS

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

The Simba Google BigQuery ODBC Connector is available for macOS as a `.dmg` file named `SimbaODBCDriverforGoogleBigQuery.dmg`. The connector supports both 32- and 64-bit client applications.

To install the Simba Google BigQuery ODBC Connector on macOS:

1. Double-click **SimbaODBCDriverforGoogleBigQuery.dmg** to mount the disk image.
2. Double-click **SimbaODBCDriverforGoogleBigQuery.pkg** to run the installer.
3. In the installer, click **Continue**.
4. On the Software License Agreement screen, click **Continue**, and when the prompt appears, click **Agree** if you agree to the terms of the License Agreement.
5. Optionally, to change the installation location, click **Change Install Location**, then select the desired location, and then click **Continue**.

   > **Note:**
   >
   > By default, the connector files are installed in the `/Library/simba/googlebigqueryodbc` directory.

6. To accept the installation location and begin the installation, click **Install**.

7. When the installation completes, click **Close**.
8. If you received a license file through email, then copy the license file into the `/lib` subfolder in the connector installation directory. You must have root privileges when changing the contents of this folder.

   For example, if you installed the connector to the default location, you would copy the license file into the `/Library/simba/googlebigqueryodbc/lib` folder.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 28.

Verifying the Connector Version Number on macOS

If you need to verify the version of the Simba Google BigQuery ODBC Connector that is installed on your macOS machine, you can query the version number through the Terminal.

To verify the connector version number on macOS:

- At the Terminal, run the following command:

```
pkgutil --info com.simba.googlebigqueryodbc
```

The command returns information about the Simba Google BigQuery ODBC Connector that is installed on your machine, including the version number.

Linux Connector

The Linux connector is available as an RPM file and as a tarball package.

Linux System Requirements

Install the connector on client machines where the application is installed. Each client machine that you install the connector on must meet the following minimum system requirements:

- One of the following distributions:
  - Red Hat® Enterprise Linux® (RHEL) 7
  - CentOS 7
  - SUSE Linux Enterprise Server (SLES) 11 or 12
  - Debian 8 or 9
  - Ubuntu 14.04, 16.04, or 18.04
- 150 MB of available disk space
- One of the following ODBC driver managers installed:
  - iODBC 3.52.9 or later
  - unixODBC 2.2.14 or later
- glibc 2.17 or later

To install the connector, you must have root access on the machine.

Installing the Connector Using the RPM File

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

On 64-bit editions of Linux, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application:

- `simbagooglebigquery-[Version]-[Release].i686.rpm` for the 32-bit connector
- `simbagooglebigquery-[Version]-[Release].x86_64.rpm` for the 64-bit connector

The placeholders in the file names are defined as follows:

- *[Version]* is the version number of the connector.
- *[Release]* is the release number for this version of the connector.

You can install both the 32-bit and 64-bit versions of the connector on the same machine.

To install the Simba Google BigQuery ODBC Connector using the RPM File:

1. Log in as the root user.
2. Navigate to the folder containing the RPM package for the connector.
3. Depending on the Linux distribution that you are using, run one of the following commands from the command line, where *[RPMFileName]* is the file name of the RPM package:
   - If you are using Red Hat Enterprise Linux or CentOS, run the following command:

     ```
     yum --nogpgcheck localinstall [RPMFileName]
     ```

   - Or, if you are using SUSE Linux Enterprise Server, run the following command:

     ```
     zypper install [RPMFileName]
     ```

   The Simba Google BigQuery ODBC Connector files are installed in the `/opt/simba/googlebigqueryodbc` directory.
4. If you received a license file through email, then copy the license file into the `/opt/simba/googlebigqueryodbc/lib/32` or `/opt/simba/googlebigqueryodbc/lib/64` folder, depending on the version of the connector that you installed.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 28.

Installing the Connector Using the Tarball Package

If you did not obtain this connector from the Simba website, you might need to follow a different installation procedure. For more information, see the *Simba OEM ODBC Connectors Installation Guide*.

The Simba Google BigQuery ODBC Connector is available as a tarball package named `SimbaODBCDriverforGoogleBigQuery_[Version].[Release]-Linux.tar.gz`, where *[Version]* is the version number of the connector and *[Release]* is the release number for this version of the connector. The package contains both the 32-bit and 64-bit versions of the connector.

On 64-bit editions of Linux, you can execute both 32- and 64-bit applications. However, 64-bit applications must use 64-bit connectors, and 32-bit applications must use 32-bit connectors. Make sure that you use a connector whose bitness matches the bitness of the client application. You can install both versions of the connector on the same machine.

To install the connector using the tarball package:

1. Log in as the root user, and then navigate to the folder containing the tarball package.
2. Run the following command to extract the package and install the connector:

   ```
   tar  --directory=/opt -zxvf [TarballName]
   ```

   Where `[TarballName]` is the name of the tarball package containing the connector.

   The Simba Google BigQuery ODBC Connector files are installed in the `opt/simba/googlebigqueryodbc` directory.
3. If you received a license file through email, then copy the license file into the `opt/simba/googlebigqueryodbc/lib/32` or `opt/simba/googlebigqueryodbc/lib/64` folder, depending on the version of the connector that you installed.

Next, configure the environment variables on your machine to make sure that the ODBC driver manager can work with the connector. For more information, see Configuring the ODBC Driver Manager on Non-Windows Machines on page 28.

Verifying the Connector Version Number on Linux

If you need to verify the version of the Simba Google BigQuery ODBC Connector that is installed on your Linux machine, you can query the version number through the command-line interface if the connector was installed using an RPM file. Alternatively, you can search the connector's binary file for version number information.

To verify the connector version number on Linux using the command-line interface:

- Depending on your package manager, at the command prompt, run one of the following commands:

  - ```
    yum list 'Simba*'| grep
    SimbaODBCDriverforGoogleBigQuery
    ```
    ```

- `rpm -qa | grep SimbaODBCDriverforGoogleBigQuery`

The command returns information about the Simba Google BigQuery ODBC Connector that is installed on your machine, including the version number.

To verify the connector version number on Linux using the binary file:

1. Navigate to the `/lib` subfolder in your connector installation directory. By default, the path to this directory is: `/opt/simba/googlebigqueryodbc/lib`.
2. Open the connector's `.so` binary file in a text editor, and search for the text `$driver_version_sb$:`. The connector's version number is listed after this text.

Configuring the ODBC Driver Manager on Non-Windows Machines

To make sure that the ODBC driver manager on your machine is configured to work with the Simba Google BigQuery ODBC Connector, do the following:

- Set the library path environment variable to make sure that your machine uses the correct ODBC driver manager. For more information, see Specifying ODBC Driver Managers on Non-Windows Machines on page 28.
- If the connector configuration files are not stored in the default locations expected by the ODBC driver manager, then set environment variables to make sure that the driver manager locates and uses those files. For more information, see Specifying the Locations of the Connector Configuration Files on page 29.

After configuring the ODBC driver manager, you can configure a connection and access your data store through the connector.

Specifying ODBC Driver Managers on Non-Windows Machines

You need to make sure that your machine uses the correct ODBC driver manager to load the connector. To do this, set the library path environment variable.

macOS

If you are using a macOS machine, then set the DYLD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set DYLD_LIBRARY_PATH for the current user session:

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/lib
```

For information about setting an environment variable permanently, refer to the macOS shell documentation.

Linux

If you are using a Linux machine, then set the LD_LIBRARY_PATH environment variable to include the paths to the ODBC driver manager libraries. For example, if the libraries are installed in `/usr/local/lib`, then run the following command to set LD_LIBRARY_PATH for the current user session:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

For information about setting an environment variable permanently, refer to the Linux shell documentation.

Specifying the Locations of the Connector Configuration Files

By default, ODBC driver managers are configured to use hidden versions of the `odbc.ini` and `odbcinst.ini` configuration files (named `.odbc.ini` and `.odbcinst.ini`) located in the home directory, as well as the `simba.googlebigqueryodbc.ini` file in the `lib` subfolder of the connector installation directory. If you store these configuration files elsewhere, then you must set the environment variables described below so that the driver manager can locate the files.

If you are using iODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.
- Set ODBCINSTINI to the full path and file name of the `odbcinst.ini` file.
- Set SIMBAGOOGLEBIGQUERYODBCINI to the full path and file name of the `simba.googlebigqueryodbc.ini` file.

If you are using unixODBC, do the following:

- Set ODBCINI to the full path and file name of the `odbc.ini` file.
- Set ODBCSYSINI to the full path of the directory that contains the `odbcinst.ini` file.
- Set SIMBAGOOGLEBIGQUERYODBCINI to the full path and file name of the `simba.googlebigqueryodbc.ini` file.

For example, if your `odbc.ini` and `odbcinst.ini` files are located in `/usr/local/odbc` and your `simba.googlebigqueryodbc.ini` file is located in `/etc`, then set the environment variables as follows:

For iODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCINSTINI=/usr/local/odbc/odbcinst.ini
export
SIMBAGOOGLEBIGQUERYODBCINI=/etc/simba.googlebigqueryodbc.ini
```

For unixODBC:

```
export ODBCINI=/usr/local/odbc/odbc.ini
export ODBCSYSINI=/usr/local/odbc
export
SIMBAGOOGLEBIGQUERYODBCINI=/etc/simba.googlebigqueryodbc.ini
```

To locate the `simba.googlebigqueryodbc.ini` file, the connector uses the following search order:

1. If the SIMBAGOOGLEBIGQUERYODBCINI environment variable is defined, then the connector searches for the file specified by the environment variable.
2. The connector searches the directory that contains the connector library files for a file named `simba.googlebigqueryodbc.ini`.
3. The connector searches the current working directory of the application for a file named `simba.googlebigqueryodbc.ini`.
4. The connector searches the home directory for a hidden file named `.simba.googlebigqueryodbc.ini` (prefixed with a period).
5. The connector searches the `/etc` directory for a file named `simba.googlebigqueryodbc.ini`.

Configuring ODBC Connections on a Non-Windows Machine

The following sections describe how to configure ODBC connections when using the Simba Google BigQuery ODBC Connector on non-Windows platforms:

Creating a Data Source Name on a Non-Windows Machine

When connecting to your data store using a DSN, you only need to configure the `odbc.ini` file. Set the properties in the `odbc.ini` file to create a DSN that specifies the connection information for your data store. For information about configuring a DSN-less connection instead, see Configuring a DSN-less Connection on a Non-Windows Machine on page 34.

If your machine is already configured to use an existing `odbc.ini` file, then update that file by adding the settings described below. Otherwise, copy the `odbc.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To create a Data Source Name on a non-Windows machine:

1. In a text editor, open the `odbc.ini` configuration file.

   > **Note:**
   >
   > If you are using a hidden copy of the `odbc.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. In the `[ODBC Data Sources]` section, add a new entry by typing a name for the DSN, an equal sign (=), and then the name of the connector.

   For example, on a macOS machine:

```
[ODBC Data Sources]
Sample DSN=Simba Google BigQuery ODBC Connector
```

As another example, for a 32-bit connector on a Linux machine:

```
[ODBC Data Sources]
Sample DSN=Simba Google BigQuery ODBC Connector 32-bit
```

3. Create a section that has the same name as your DSN, and then specify configuration options as key-value pairs in the section:
   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

      For example, on a macOS machine:

      ```
      Driver=/Library/
      simba/googlebigqueryodbc/lib/libgooglebigqueryodbc_
      sbu.dylib
      ```

      As another example, for a 32-bit connector on a Linux machine:

      ```
      Driver=/opt/
      simba
      /googlebigqueryodbc/lib/32/libgooglebigqueryodbc_
      sb32.so
      ```

   b. Set the `Catalog` property to the name of your BigQuery project. This project is the default project that the Simba Google BigQuery ODBC Connector queries against, and also the project that is billed for queries that are run using this DSN.

      For example:

      ```
      Catalog=testdata
      ```

   c. Configure authentication using a Google user account or a Google service account. For more information, see Configuring Authentication on a Non-Windows Machine on page 36.
   d. Optionally, to use trusted CA certificates from a `.pem` file other than the default file installed with the connector, set the `TrustedCerts` property to the full path of the file.
   e. Optionally, to allow the connector to access Google Drive so that it can support federated tables that combine BigQuery data with data from Google Drive, set the `RequestGoogleDriveScope` property to `1`.

f. Optionally, set additional key-value pairs as needed to specify other optional connection settings. For detailed information about all the configuration options supported by the Simba Google BigQuery ODBC Connector, see Connector Configuration Options on page 59.

4. Save the `odbc.ini` configuration file.

> **Note:**
>
> If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINI environment variable specifies the location. For more information, see Specifying the Locations of the Connector Configuration Files on page 29.

For example, the following is an `odbc.ini` configuration file for macOS containing a DSN that connects to Google BigQuery using a refresh token obtained from a user account:

```
[ODBC Data Sources]
Sample DSN=Simba Google BigQuery ODBC Connector
[Sample DSN]
Driver=/Library/
simba/googlebigqueryodbc/lib/libgooglebigqueryodbc_sbu.dylib
Catalog=testdata
OAuthMechanism=1
RefreshToken=CH01pcNn/qFcYwUlJpkF_yyufYrqj4O4g7cdXvGgs-zT6
```

As another example, the following is an `odbc.ini` configuration file for a 32-bit connector on a Linux machine, containing a DSN that connects to Google BigQuery using a refresh token obtained from a user account:

```
[ODBC Data Sources]
Sample DSN=Simba Google BigQuery ODBC Connector 32-bit
[Sample DSN]
Driver=/opt/
simba/googlebigqueryodbc/lib/32/libgooglebigqueryodbc_
sb32.so
Catalog=testdata
OAuthMechanism=1
RefreshToken=CH01pcNn/qFcYwUlJpkF_yyufYrqj4O4g7cdXvGgs-zT6
```

You can now use the DSN in an application to connect to the data store.

Configuring a DSN-less Connection on a Non-Windows Machine

To connect to your data store through a DSN-less connection, you need to define the connector in the `odbcinst.ini` file and then provide a DSN-less connection string in your application.

If your machine is already configured to use an existing `odbcinst.ini` file, then update that file by adding the settings described below. Otherwise, copy the `odbcinst.ini` file from the `Setup` subfolder in the connector installation directory to the home directory, and then update the file as described below.

To define a connector on a non-Windows machine:

1. In a text editor, open the `odbcinst.ini` configuration file.

   > **Note:**
   >
   > If you are using a hidden copy of the `odbcinst.ini` file, you can remove the period (`.`) from the start of the file name to make the file visible while you are editing it.

2. In the `[ODBC Drivers]` section, add a new entry by typing a name for the connector, an equal sign (=), and then `Installed`.

   For example:

   ```
   [ODBC Drivers]
   Simba Google BigQuery ODBC Connector=Installed
   ```

3. Create a section that has the same name as the connector (as specified in the previous step), and then specify the following configuration options as key-value pairs in the section:
   a. Set the `Driver` property to the full path of the connector library file that matches the bitness of the application.

      For example, on a macOS machine:

      ```
      Driver=/Library/
      simba/googlebigqueryodbc/lib/libgooglebigqueryodbc_
      sbu.dylib
      ```

      As another example, for a 32-bit connector on a Linux machine:

      ```
      Driver=/opt/
      simba
      ```

```
/googlebigqueryodbc/lib/32/libgooglebigqueryodbc_
sb32.so
```

b. Optionally, set the `Description` property to a description of the connector.

   For example:

```
Description=Simba Google BigQuery ODBC Connector
```

4. Save the `odbcinst.ini` configuration file.

> **Note:**
>
> If you are storing this file in its default location in the home directory, then prefix the file name with a period (`.`) so that the file becomes hidden. If you are storing this file in another location, then save it as a non-hidden file (without the prefix), and make sure that the ODBCINSTINI or ODBCSYSINI environment variable specifies the location. For more information, see Specifying the Locations of the Connector Configuration Files on page 29.

For example, the following is an `odbcinst.ini` configuration file for macOS:

```
[ODBC Drivers]
Simba Google BigQuery ODBC Connector=Installed
[Simba Google BigQuery ODBC Connector]
Description=Simba Google BigQuery ODBC Connector
Driver=/Library/
simba/googlebigqueryodbc/lib/libgooglebigqueryodbc_sbu.dylib
```

As another example, the following is an `odbcinst.ini` configuration file for both the 32- and 64-bit connectors on Linux:

```
[ODBC Drivers]
Simba Google BigQuery ODBC Connector 32-bit=Installed
Simba Google BigQuery ODBC Connector 64-bit=Installed
[Simba Google BigQuery ODBC Connector 32-bit]
Description=Simba Google BigQuery ODBC Connector (32-bit)
Driver=/opt/
simba/googlebigqueryodbc/lib/32/libgooglebigqueryodbc_
sb32.so
[Simba Google BigQuery ODBC Connector 64-bit]
Description=Simba Google BigQuery ODBC Connector (64-bit)
```

```
Driver=/opt/
simba/googlebigqueryodbc/lib/64/libgooglebigqueryodbc_
sb64.so
```

You can now connect to your data store by providing your application with a connection string where the `Driver` property is set to the connector name specified in the `odbcinst.ini` file, and all the other necessary connection properties are also set. For more information, see "DSN-less Connection String Examples" in Using a Connection String on page 45.

For instructions about configuring specific connection features, see the following:

- Configuring Authentication on a Non-Windows Machine on page 36
- Configuring the High-Throughput API on a Non-Windows Machine on page 40

For detailed information about all the connection properties that the connector supports, see Connector Configuration Options on page 59.

Configuring Authentication on a Non-Windows Machine

The Simba Google BigQuery ODBC Connector uses the OAuth 2.0 protocol for authentication and authorization. It authenticates your connection through Google OAuth APIs. You can configure the connector to provide your credentials and authenticate the connection to the database using one of the following methods:

- Using a Google User Account on page 36
- Using a Google Service Account on page 39

Using a Google User Account

You can configure the connector to authenticate the connection with a Google user account. This authentication method uses the OAuth 2.0 access and refresh tokens associated with the user account as the credentials.

The access token is transmitted with every API call that the connector makes, and it is required for accessing BigQuery data stores. However, the access token expires after a certain amount of time and must be renewed using the refresh token. If the refresh token is stored in the DSN, the connector automatically uses it to renew access tokens when they expire.

Note:

For more information about OAuth 2.0, see "Using OAuth 2.0 to Access Google APIs" in the Google Identity Platform documentation: https://developers.google.com/identity/protocols/OAuth2.

At minimum, you need to provide the OAuth 2.0 refresh token associated with your account. The connector retrieves and uses an access token based on your specified refresh token.

- If you do not have your refresh token, see Retrieving a Refresh Token on page 37.
- If you have your refresh token, see Providing a Refresh Token on page 38.
- If you want to provide a `.json` key file that contains your credentials instead of providing your refresh token directly in your connection information, see Providing a Key File on page 39.

Retrieving a Refresh Token

When you authenticate your connection this way, the authentication process provides a temporary authorization code that you can exchange for an access token and a refresh token.

You can retrieve a refresh token by providing your own credentials, or by using a script that uses Simba-provided credentials.

Note:

If you use your credentials to generate a refresh token, you cannot use it in conjunction with the Simba-provided credentials. Conversely, if you use a refresh token generated with the Simba-provided credentials, it cannot be used in conjunction with your user credentials.

To configure authentication by retrieving a refresh token using Simba-provided credentials on a non-Windows machine:

1. In the *[INSTALL_DIR]*`/Tools` directory, right-click `get_refresh_token.sh` and select **Edit**.
2. From the internal commented section, copy the entire authentication generator URL.
3. In a web browser, navigate to the URL you copied in the previous step.
4. Click **Allow**.

   The browser redirects you to a page with an authentication token.

5. Copy the authentication token.
6. Using a command line interface, run `get_refresh_token.sh` with your copied authentication token added as the argument to the script.

   The script generates a refresh token.

Now that you have a refresh token, see Providing a Refresh Token on page 38.

To configure user account authentication by retrieving a refresh token on a non-Windows machine:

1. Obtain a refresh token based on your user account:
   a. In a web browser, navigate to the Google OAuth 2.0 Playground: https://developers.google.com/oauthplayground/.
   b. In the side panel, expand **BigQuery API v2** and select the appropriate scope for the permissions that you need.

      > **Note:**
      >
      > For information about the permissions associated with each scope, see "OAuth 2.0 Scopes for Google APIs" in the Google Identity Platform documentation: https://developers.google.com/identity/protocols/googlescopes.

   c. Click **Authorize APIs**.
   d. Sign in to your user account.
   e. When you are prompted to allow Google OAuth 2.0 Playground to view and manage your data in Google BigQuery, click **Allow**.

      The authentication process returns you to the Google OAuth 2.0 Playground, and automatically populates the Authorization Code field with an authorization code.
   f. Click **Exchange Authorization Code for Tokens**.

      The Refresh Token and Access Token fields are populated with the appropriate token values.
2. In your DSN or connection string, set the `OAuthMechanism` property to `1`.
3. Set the `RefreshToken` property to the refresh token that you obtained from Google.
4. Set the `ClientId` property to your BigQuery client ID.
5. Set the `ClientSecret` property to the corresponding client secret.

Providing a Refresh Token

If you already have your refresh token, then you can provide the token in your connection information without going through the retrieval process described above.

To configure user account authentication by providing a refresh token on a non-Windows machine:

1. Set the `OAuthMechanism` property to `1`.
2. Set the `RefreshToken` property to the refresh token associated with your user account.

Providing a Key File

As an alternative to providing your refresh token directly in your connection information, you can save the token in a `.json` key file and then specify the path to the file in your connection information.

The file must define a JSON object of type `authorized_user` containing the refresh token, client ID, and client secret associated with your user account. For example, the `.json` key file must be written in the following format:

```
{
   "type": "authorized_user",
   "client_id": "[YourClientID]",
   "client_secret":"[YourClientSecret]",
   "refresh_token":"[YourRefreshToken]"
}
```

To configure user account authentication by providing a key file on a non-Windows machine:

1. Set the `OAuthMechanism` property to `0`.

   Note:

   Although this is a form of user authentication, the connector must be configured to use the service authentication mechansim (`OAuthMechanism=0`) in order to detect and use the key file.

2. Set the `Email` property to your user account email ID.
3. Set the `KeyFilePath` property to the full path to the `.json` key file.

Using a Google Service Account

You can configure the connector to authenticate the connection with a Google service account. When you authenticate your connection this way, the connector handles authentication on behalf of the service account, so that an individual user account is not directly involved and no user input is required.

To authenticate your connection this way, you must provide a Google service account email address and the full path to a private key file for the service account. You can generate and download the private key file when you set up the service account.

Note:

- For more information about OAuth 2.0 authentication using a service account, see "Using OAuth 2.0 for Server to Server Applications" in the Google Identity Platform documentation: https://developers.google.com/identity/protocols/OAuth2ServiceAccount.
- For information about obtaining service account keys, see "Creating and Managing Service Account Keys" in the Google Cloud Identity & Access Management documentation: https://cloud.google.com/iam/docs/creating-managing-service-account-keys.

To configure service account authentication on a non-Windows machine:

1. Set the `OAuthMechanism` property to `0`.
2. Set the `Email` property to your service account email ID.
3. Set the `KeyFilePath` property to the full path to the `.p12` or `.json` key file that is used to authenticate the service account ID.

Configuring the High-Throughput API on a Non-Windows Machine

You can configure the connector to use the High-Throughput API to handle large result sets more efficiently. For more information about the High-Throughput API, see High-Throughput API on page 55.

To configure the High-Throughput API on a non-Windows machine:

1. Make sure that your Google BigQuery project has the Storage API enabled. For more information about the Storage API, see "BigQuery Storage API Overview" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/reference/storage/.

   Note:

   On Linux, if the application running the connector has been built with an RPATH, then you must either include the connector directory in that RPATH, or adjust your LD_LIBRARY_PATH to include the connector directory.

2. In the `odbc.ini` file, if you are using Legacy SQL (the `SQLDialect` property is set to `0`), then set the `AllowLargeResults` property to `1`.

3. To specify the dataset that stores temporary tables for large result sets and result sets returned by the High-Throughput API, do one of the following:
   - To use the default dataset with the ID _bqodbc_temp_tables, set the `UseDefaultLargeResultsDataset` property to `1`.
   - Or, to specify a different dataset, set the `UseDefaultLargeResultsDataset` property to `0` and set the `LargeResultsDataSetId` property to the ID of the BigQuery dataset that you want to use.

   Note:

   If the dataset does not exist and the data store specifies the US region, the connector creates the dataset.
4. Set the `EnableHTAPI` property to `1`.
5. Set the `HTAPI_MinResultsSize` property to the minimum number of table rows required to activate the High-Throughput API.
6. Set the `HTAPI_MinActivationRatio` property to the minimum ratio of total rows to rows in the first page required to activate reading through the High-Throughput API.

   Note:

   If this value is set to 0, then the connector uses the High-Throughput API for all query results that meet the minimum results size specified by the `HTAPI_MinResultsSize` property.

The connector uses the BigQuery High-Throughput API instead of the REST API for requests where both:

- the number of table rows in your query results exceeds the `HTAPI_MinResultsSize` value;
- and the number of pages in the results exceeds the `HTAPI_MinActivationRatio` value.

Configuring Logging Options on a Non-Windows Machine

To help troubleshoot issues, you can enable logging in the connector.

Important:

Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.

You can set the connection properties described below in a connection string, in a DSN (in the `odbc.ini` file), or as a connector-wide setting (in the `simba.googlebigqueryodbc.ini` file). Settings in the connection string take precedence over settings in the DSN, and settings in the DSN take precedence over connector-wide settings.

To enable logging on a non-Windows machine:

1. To specify the level of information to include in log files, set the `LogLevel` property to one of the following numbers:

LogLevel Value	Description
0	Disables all logging.
1	Logs severe error events that lead the connector to abort.
2	Logs error events that might allow the connector to continue running.
3	Logs events that might result in an error if action is not taken.
4	Logs general information that describes the progress of the connector.
5	Logs detailed information that is useful for debugging the connector.
6	Logs all connector activity.

2. Set the `LogPath` key to the full path to the folder where you want to save log files.
3. Set the `LogFileCount` key to the maximum number of log files to keep.

> **Note:**
>
> After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

4. Set the `LogFileSize` key to the maximum size of each log file in bytes.

> **Note:**
>
> After the maximum file size is reached, the connector creates a new file and continues logging.

5. Save the `simba.googlebigqueryodbc.ini` configuration file.
6. Restart your ODBC application to make sure that the new settings take effect.

The Simba Google BigQuery ODBC Connector produces the following log files at the location you specify using the `LogPath` key:

- A `simbabigqueryodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `simbabigqueryodbcdriver_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

To disable logging on a non-Windows machine:

1. Set the `LogLevel` key to `0`.
2. Save the `simba.googlebigqueryodbc.ini` configuration file.
3. Restart your ODBC application to make sure that the new settings take effect.

Testing the Connection on a Non-Windows Machine

To test the connection, you can use an ODBC-enabled client application. For a basic connection test, you can also use the test utilities that are packaged with your driver manager installation. For example, the iODBC driver manager includes simple utilities called iodbctest and iodbctestw. Similarly, the unixODBC driver manager includes simple utilities called isql and iusql.

Using the iODBC Driver Manager

You can use the iodbctest and iodbctestw utilities to establish a test connection with your connector. Use iodbctest to test how your connector works with an ANSI application, or use iodbctestw to test how your connector works with a Unicode application.

> **Note:**
>
> There are 32-bit and 64-bit installations of the iODBC driver manager available. If you have only one or the other installed, then the appropriate version of iodbctest (or iodbctestw) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the iODBC driver manager, see http://www.iodbc.org.

To test your connection using the iODBC driver manager:

1. Run **iodbctest** or **iodbctestw**.
2. Optionally, if you do not remember the DSN, then type a question mark (`?`) to see a list of available DSNs.
3. Type the connection string for connecting to your data store, and then press ENTER. For more information, see Using a Connection String on page 45.

If the connection is successful, then the `SQL>` prompt appears.

Using the unixODBC Driver Manager

You can use the isql and iusql utilities to establish a test connection with your connector and your DSN. isql and iusql can only be used to test connections that use a DSN. Use isql to test how your connector works with an ANSI application, or use iusql to test how your connector works with a Unicode application.

> **Note:**
>
> There are 32-bit and 64-bit installations of the unixODBC driver manager available. If you have only one or the other installed, then the appropriate version of isql (or iusql) is available. However, if you have both 32- and 64-bit versions installed, then you need to make sure that you are running the version from the correct installation directory.

For more information about using the unixODBC driver manager, see http://www.unixodbc.org.

To test your connection using the unixODBC driver manager:

- Run isql or iusql by using the corresponding syntax:
  - `isql` *`[DataSourceName]`*
  - `iusql` *`[DataSourceName]`*

  `[DataSourceName]` is the DSN that you are using for the connection.

If the connection is successful, then the `SQL>` prompt appears.

> **Note:**
>
> For information about the available options, run isql or iusql without providing a DSN.

Using a Connection String

For some applications, you might need to use a connection string to connect to your data source. For detailed information about how to use a connection string in an ODBC application, refer to the documentation for the application that you are using.

The connection strings in the following sections are examples showing the minimum set of connection attributes that you must specify to successfully connect to the data source. Depending on the configuration of the data source and the type of connection you are working with, you might need to specify additional connection attributes. For detailed information about all the attributes that you can use in the connection string, see Connector Configuration Options on page 59.

DSN Connection String Example

The following is an example of a connection string for a connection that uses a DSN:

```
DSN=[DataSourceName]
```

[DataSourceName] is the DSN that you are using for the connection.

You can set additional configuration options by appending key-value pairs to the connection string. Configuration options that are passed in using a connection string take precedence over configuration options that are set in the DSN.

DSN-less Connection String Examples

Some applications provide support for connecting to a data source using a connector without a DSN. To connect to a data source without using a DSN, use a connection string instead.

The placeholders in the examples are defined as follows, in alphabetical order:

- *[PortNumber]* is the number of the TCP port that the proxy server uses to listen for client connections.
- *[Project]* is the BigQuery project containing the data that you want to use.
- *[Server]* is the IP address or host name of the proxy server to which you are connecting.
- *[ServiceAccount]* is your service account email ID.
- *[ServiceKeyPath]* is the full path to a `.p12` or `.json` key file for service account authentication.

- *[Token]* is the refresh token that you obtain from Google for authorizing access to BigQuery.
- *[UserAccount]* is your user account email ID.
- *[UserKeyPath]* is the full path to a `.json` key file containing your refresh token, client ID, and client secret. For information about the required format of the `.json` file, see Key File Path on page 64.

Connecting to Google BigQuery using a User Account

The following is the format of a DSN-less connection string for a user account connection to Google BigQuery:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=1;RefreshToken=[Token];Catalog=[Project];
```

For example:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=1;RefreshToken=CH01pcNn/qFcYwUlJpkF_
yyufYrqj4O4g7cdXvGgs-zT6;Catalog=testdata;
```

As an alternative to providing your refresh token directly in the string, you can save your credentials in a `.json` key file and then provide the full path to that file in your string. In this case, the connection string must be written in the following format:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=0;Email=[UserAccount];KeyFilePath=
[UserKeyPath];Catalog=[Project];
```

For example:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=0;Email=simba@gmail.com;
KeyFilePath=C:\SecureFiles\UserKeyFile.json;Catalog=testdat
a;
```

Connecting to Google BigQuery using a Service Account

The following is the format of a DSN-less connection string for a service account connection to Google BigQuery:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=0;Email=[ServiceAccount];KeyFilePath=
[ServiceKeyPath];Catalog=[Project];
```

For example:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=0;Email=application-service-
account@iam.gserviceaccount.com;KeyFilePath=C:\SecureFiles\S
erviceKeyFile.p12;Catalog=testdata;
```

Connecting to Google BigQuery through a Proxy Server

The following is the format of a DSN-less connection string for connecting to Google BigQuery with a user account through a proxy server:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=1;RefreshToken=[Token];Catalog=[Project];
ProxyHost=[Server];ProxyPort=[PortNumber];
```

For example:

```
Driver=Simba Google BigQuery ODBC Connector;
OAuthMechanism=1;RefreshToken=CH01pcNn/qFcYwUlJpkF_
yyufYrqj4O4g7cdXvGgs-zT6;
Catalog=testdata;ProxyHost=192.168.222.160;
ProxyPort=8000;
```

Features

For more information on the features of the Simba Google BigQuery ODBC Connector, see the following:

Data Types

The Simba Google BigQuery ODBC Connector supports many common data formats, converting between BigQuery data types and SQL data types.

- Data type mappings: BigQuery to SQL
- Data type mappings: SQL to BigQuery

The following table lists the supported data type mappings from BigQuery to SQL.

BigQuery Data Type	SQL Data Type
ARRAY	SQL_VARCHAR

BigQuery Data Type	SQL Data Type
BIGNUMERIC	SQL_NUMERIC
	SQL_DECIMAL **Note:** The connector sends SQL_DECIMAL data to BigQuery as BIGNUMERIC data, because BigQuery does not support a DECIMAL data type. The connector always returns BIGNUMERIC data as SQL_NUMERIC data, and sends SQL_NUMERIC data to BigQuery as BIGNUMERIC data.
BOOL	SQL_BIT
BOOLEAN	SQL_BIT
BYTES	SQL_VARBINARY
DATE	SQL_DATE
DATETIME	SQL_TYPE_TIMESTAMP **Note:** For ODBC versions prior to ODBC 3, the connector uses SQL_TIMESTAMP.
FLOAT64	SQL_DOUBLE

<table>
<tr><th>BigQuery Data Type</th><th>SQL Data Type</th></tr>
<tr><td>GEOGRAPHY</td><td>SQL_VARCHAR or SQL_WVARCHAR.
Note:
For information about whether GEOGRAPHY data is returned as SQL_VARCHAR or SQL_WVARCHAR, see Use SQL_WVARCHAR instead of SQL_VARCHAR on page 75.
Geography data can only be inserted using specific functions. For more information, see: https://cloud.google.com/bigquery/docs/reference/standard-sql/geography_functions.</td></tr>
<tr><td>INTEGER</td><td>SQL_BIGINT</td></tr>
<tr><td>INT64</td><td>SQL_BIGINT</td></tr>
<tr><td rowspan="2">NUMERIC</td><td>SQL_NUMERIC</td></tr>
<tr><td>SQL_DECIMAL
Note:
The connector sends SQL_DECIMAL data to BigQuery as NUMERIC data, because BigQuery does not support a DECIMAL data type.
The connector always returns NUMERIC data as SQL_NUMERIC data, and sends SQL_NUMERIC data to BigQuery as NUMERIC data.</td></tr>
<tr><td>STRING</td><td>SQL_VARCHAR or SQL_WVARCHAR
Note:
For information about whether STRING data is returned as SQL_VARCHAR or SQL_WVARCHAR, see Use SQL_WVARCHAR instead of SQL_VARCHAR on page 75.</td></tr>
</table>

BigQuery Data Type	SQL Data Type
STRUCT	SQL_VARCHAR
TIME	SQL_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP **Note:** For ODBC versions prior to ODBC 3, the connector uses SQL_TIMESTAMP.

The following table lists the supported data type mappings from SQL to BigQuery.

SQL Data Type	BigQuery Data Type
SQL_BIGINT	INT64
SQL_BIT	BOOL
SQL_CHAR	STRING
SQL_DATE	DATE
SQL_DECIMAL	NUMERIC
SQL_DOUBLE	FLOAT64
SQL_INTEGER	INT64
SQL_LONGVARBINARY	BYTES
SQL_LONGVARCHAR	STRING
SQL_NUMERIC	NUMERIC
SQL_SMALLINT	INT64
SQL_TIME	TIME

SQL Data Type	BigQuery Data Type
SQL_TIMESTAMP	TIMESTAMP
SQL_TINYINT	INT64
SQL_TYPE_DATE	DATE
SQL_TYPE_TIME	TIME
SQL_TYPE_TIMESTAMP	TIMESTAMP
SQL_VARBINARY	BYTES
SQL_VARCHAR	STRING
SQL_WLONGVARCHAR	STRING
SQL_WVARCHAR	STRING

Nested and Repeated Records

The Simba Google BigQuery ODBC Connector partially supports nested and repeated records.

In Standard SQL, the record is returned as a BigQuery string, and converted to SQL_VARCHAR. The Standard SQL syntax represents the sub-components of record data as nested sub-types. In the example below, `city` and `years` belong to the base record type of `address`, and `name` does not.

```
{
   "v": { // "column" object
      "f": [
         { // "address" value
            "v": [ // "address" array
               { // "city" value
                  "v": "Vancouver"
               },
               { // "years" value
                  "v": "5"
               }
            ]
         },
```

```
            { // "name" value
               "v": "Google"
            }
        ]
    }
}
```

In Legacy SQL, the record is flattened by BigQuery before it is returned by the connector. The query returns the data as the schema of a flat table. All STRUCT members are their own columns, and all ARRAY members are expanded into as many rows as there are array elements.

Arrays

The Simba Google BigQuery ODBC Connector fully supports ARRAY data types. The connector returns the base ARRAY type as a text representation of the JSON array object.

For example, the SQL statement SELECT [1,2,3] returns the following JSON:

```
{
    "v":[
        {
            "v":"1",
        },
        {
            "v":"2",
        },
        {
            "v":"3"
        }
    ]
}
```

Security and Authentication

To protect data from unauthorized access, BigQuery data stores require all connections to be authenticated using the OAuth 2.0 protocol and encrypted using TLS 1.2 with one-way authentication. The Simba Google BigQuery ODBC Connector protects your data by providing support for these authentication protocols and further obscuring data from unwanted access by fetching it in a non-text format. The data is compressed using zlib and encrypted using TLS.

The connector provides mechanisms that allow you to complete an OAuth 2.0 authentication flow using a Google user account or a Google service account. The connector retrieves a token based on the account credentials specified in your DSN or connection string, and then uses the token to authenticate the connection to BigQuery. For detailed configuration instructions, see Configuring Authentication on Windows on page 11 or Configuring Authentication on a Non-Windows Machine on page 36.

Additionally, the connector automatically encrypts all connections with TLS. TLS encryption protects data and credentials when they are transferred over the network, and provides stronger security than authentication alone. By default, the connector uses the trusted CA certificates file that is included during installation, but you can configure the connector to use a different file by setting the Trusted Certificates option (the `TrustedCerts` property). On Windows machines, you can configure the connector to use the system trust store by setting the Use System Trust Store option (the `UseSystemTrustStore` property). For detailed configuration instructions, see Creating a Data Source Name on Windows on page 9 or Creating a Data Source Name on a Non-Windows Machine on page 31.

Catalog and Schema Support

The Simba Google BigQuery ODBC Connector supports both catalogs and schemas to make it easy for the connector to work with various ODBC applications. Projects are mapped to catalogs, and table datasets are mapped to schemas. For more information, see Catalog (Project) on page 61.

Large Result Set Support

The Simba Google BigQuery ODBC Connector supports the `AllowLargeResults` option in BigQuery job configurations, enabling result sets greater than 128MB (compressed). To store large query results, the connector creates temporary tables in BigQuery under the dataset ID specified using the Dataset Name For Large Result Sets connector configuration property. These temporary tables exist for a limited time, specified using the Temporary Table Expiration Time connector configuration property, before they are deleted.

Large result sets are always supported if Standard SQL is used. If Legacy SQL is used, large result sets are only supported if the Allow Large Result Sets check box is selected or the `AllowLargeResults` option is set to `1`.

For more information about large result sets and the limitations of this feature, see the following sections in the BigQuery documentation:

- "Queries" in *Quota Policy*: https://developers.google.com/bigquery/quota-policy.
- "Returning large query results" in *Query Data*: https://developers.google.com/bigquery/querying-data.

Note:

To enable the High-Throughput API for large result sets, select the Enable High-Throughput API check box or set the `EnableHTAPI` property to `1`. For more information, see High-Throughput API on page 55.

High-Throughput API

The High-Throughput API is a new feature of the Simba Google BigQuery ODBC Connector. This API enables the connector to leverage the BigQuery Storage API, allowing higher data throughput than the standard REST API. With this API, the connector can handle large result sets more efficiently. For more information about the Storage API, see "BigQuery Storage API Overview" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/reference/storage/.

The connector checks the number of rows in an incoming result set table and extrapolates the number of pages needed to retrieve all the results. If the number of table rows and the ratio of rows to pages both exceed the defined thresholds, the connector uses the BigQuery Storage API. Should the connector encounters any issues with initializing the storage API for retrieval, it falls back to using the standard REST API, unless this is a permissions issue.

You can customize the thresholds for using the BigQuery Storage API. For information about the configuration options used to determine when the API is used, see the following:

- Enable High-Throughput API on page 64
- Minimum Query Results Size for HTAPI on page 68
- Ratio of Results to Rows Per Block on page 71

Requirements

Before you can use the High-Throughput API, you must make certain that your system meets the following requirements:

- The BigQuery project that you are querying must have the BigQuery Storage API enabled. For more information, see "Enabling the API" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/reference/storage/#enabling_the_api.

> **Important:**
>
> Pricing for the BigQuery Storage API is different than pricing for the standard API. For more information, see "BigQuery Storage API Pricing" in the Google BigQuery documentation: https://cloud.google.com/bigquery/pricing#storage-api.

- On Windows, the connector must not be configured to use the trust store, that is, the Use System Trust Store check box must not be selected or the `UseSystemTrustStore` property must not be set to `1`.
- To enable the High-Throughput API for use with large result sets, the Enable High-Throughput API check box must be selected or the `EnableHTAPI` property must be set to `1`.

Write-Back

The Simba Google BigQuery ODBC Connector supports Data Manipulation Language (DML) statements such as INSERT, MERGE, and DELETE.

For example, the following INSERT statement is supported:

```
INSERT INTO MyTable (Col1, Col2) VALUES ("Key", "Value");
```

The connector also supports Data Definition Language (DDL) statements. Be aware that BigQuery supports specific syntax for DDL statements, and your statements must be written in that syntax. For more information, see "Using Data Definition Language Statements" in Google BigQuery's *Standard SQL Query Reference*: https://cloud.google.com/bigquery/docs/data-definition-language.

Positional Parameters

A parameterized query contains placeholders that are used for parameters. The values of those parameters are supplied at execution time.

The Simba Google BigQuery ODBC Connector supports SQL positional parameters. Parameters are specified in queries with a question mark (`?`).

For example, the following parameterized query is supported:

```
SELECT * FROM MyTable WHERE Col1=?
```

ODBC Escapes

The Simba Google BigQuery ODBC Connector supports a subset of the ODBC escape syntaxes. For a complete list of the escapes that the connector supports, call SQLGetInfo from the connector.

For more information about ODBC escapes, see "ODBC Escape Sequences" in the Programmer's Reference: https://msdn.microsoft.com/en-us/library/ms711838(v=vs.85).aspx.

For information about known issues that occur for specific ODBC escape use cases, see the "Known Issues" section in the *Magnitude Simba Google BigQuery ODBC Data Connector Release Notes*.

Service Endpoints

The Simba Google BigQuery ODBC Connector uses the following API service endpoints:

API Service	Service Endpoint
Default server name	`www.googleapis.com`
Default API URL	`https://www.googleapis.com/bigquery/v2`
Default access token request URL (User authentication)	`https://oauth2.googleapis.com/token`
Default access token request URL (Service authentication)	`https://accounts.google.com/o/oauth2/auth?`
Default OAuth scope	`https://www.googleapis.com/auth/cloud-platform`
Google Drive scope	`https://www.googleapis.com/auth/drive`

API Service	Service Endpoint
Google Client X509 certification URL	`https://www.googleapis.com/robot/v1/metadata/x509/`
Storage API URL	`https://bigquerystorage.googleapis.com/`

In addition, if you are behind a proxy, you must make sure that the proxy allows communication to `crl.pki.goog` and `ocsp.pki.goog`.

Connector Configuration Options

Connector Configuration Options lists the configuration options available in the Simba Google BigQuery ODBC Connector alphabetically by field or button label. Options having only key names, that is, not appearing in the user interface of the connector, are listed alphabetically by key name.

When creating or configuring a connection from a Windows machine, the fields and buttons described below are available in the following dialog boxes:

- Simba Google BigQuery ODBC Connector DSN Setup
- Advanced Options
- Logging Options

When using a connection string or configuring a connection from a non-Windows machine, use the key names provided below.

Configuration Options Appearing in the User Interface

The following configuration options are accessible via the Windows user interface for the Simba Google BigQuery ODBC Connector, or via the key name when using a connection string or configuring a connection from a Linux or macOS computer:

Additional Projects

Key Name	Default Value	Required
`AdditionalProjects`	None	No

Description

A comma-separated list of public BigQuery projects that the connector can access and use as catalogs. These projects are available as catalogs in metadata functions.

Allow Large Result Sets

Key Name	Default Value	Required
`AllowLargeResults`	Clear (`0`)	No

Description

This option specifies the connector's response to query results larger than 128MB when using Legacy SQL.

- Enabled (`1`): The connector allows query results that are larger than 128MB in size.
- Disabled (`0`): The connector returns an error when query results are larger than 128MB in size.

Note:

If this option is enabled, you must also specify a dataset for storing the temporary tables. For more information see Dataset Name For Large Result Sets on page 62.

For additional information on SQL see Language Dialect on page 65.

Catalog (Project)

Key Name	Default Value	Required
`Catalog`	None	Yes

Description

The name of your BigQuery project. This project is the default project that the Simba Google BigQuery ODBC Connector queries against, and is also the project that is billed for queries that are run using the DSN.

Simba ODBC Connector for Google BigQuery supports multiple catalogs, the equivalent of Google BigQuery projects.

For queries, tables in the projection list must be fully qualified, in the format of `catalog.schema.table`. If the catalog is not specified, the connector will assume the project specified by the **projectId** connection option.

For catalog functions, to retrieve information from the desired catalog, the ODBC **SQLSetConnectAttr** method must be called with **SQL_ATTR_CURRENT_CATALOG** set to the desired catalog.

Note:

To use the High-Throughput API, the specified project must have the BigQuery Storage API enabled. For more information, see "Enabling the API" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/reference/storage/#enabling_the_api.

Confirmation Code

Key Name	Default Value	Required
N/A	None	No

Description

The code that you obtain from Google for generating a refresh token.

> **Note:**
>
> The confirmation code can only be used once. You must get a new confirmation code from Google whenever you need another refresh token.

> **Note:**
>
> This option is only available on Windows.

Dataset Name For Large Result Sets

Key Name	Default Value	Required
`LargeResultsDataSetId`	None (but see Use Default _bqodbc_ temp_tables Large Results Dataset on page 74)	No

Description

The ID of the BigQuery dataset that you want to use to store temporary tables for large result sets. Only specify a value for this option if you want to enable support for large result sets.

> **Note:**
>
> - If the `UseDefaultLargeResultsDataset` property is enabled, it overrides this value.
> - If this dataset does not exist and the data store specifies the US region, the connector creates the dataset.
> - If you want to use large result sets with Legacy SQL, you must also enable the **Allow Large Result Sets** option. See Allow Large Result Sets on page 60.

Dataset

Key Name	Default Value	Required
DefaultDataset	None	No

Description

The name of a dataset that the connector queries by default.

Specifying a default dataset enables you to use unqualified table names in SQL statements. The connector treats unqualified tables as part of the default dataset. Additionally, it treats the default dataset as part of the project that is being billed. For information about specifying the project to bill, see Catalog (Project) on page 61.

Default String Column Length

Key Name	Default Value	Required
DefaultStringColumnLength	16384	No

Description

The maximum number of characters that can be contained in STRING columns.

Email

Key Name	Default Value	Required
Email	None	Yes, if OAuth Mechanism is set to Service Authentication (OAuthMechanism=0).

Description

When configuring Service Authentication, set this option to the service account email ID.

When configuring User Authentication with a `.json` key file, set this option to your user account email ID.

Enable High-Throughput API

Key Name	Default Value	Required
`EnableHTAPI`	Clear (`0`)	No

Description

This option specifies whether the connector uses the BigQuery High-Throughput API for large result sets. For detailed information about the High-Throughput API, see High-Throughput API on page 55.

- Enabled (`1`): The connector uses the High-Throughput API for large result sets that exceed the activation ratio.
- Disabled (`0`): The connector does not use the High-Throughput API for large result sets.

Important:

Pricing for the High-Throughput API is different than pricing for the standard REST API. For more information, see "BigQuery Storage API Pricing" in the Google BigQuery documentation: https://cloud.google.com/bigquery/pricing#storage-api.

Key File Path

Key Name	Default Value	Required
`KeyFilePath`	None	Yes, if OAuth Mechanism is set to Service Authentication (`OAuthMechanism=0`).

Description

When configuring Service Authentication, set this option to the full path to the `.p12` or `.json` key file that is used to authenticate the service account email address.

When configuring User Authentication with a `.json` key file, set this option to the full path to the `.json` key file containing your OAuth 2.0 credentials. The file must define a

JSON object of type `authorized_user` containing the refresh token, client ID, and client secret associated with your user account. For example, the `.json` key file must be written in the following format:

```
{
   "type": "authorized_user",
   "client_id": "[YourClientID]",
   "client_secret":"[YourClientSecret]",
   "refresh_token":"[YourRefreshToken]"
}
```

Language Dialect

Key Name	Default Value	Required
`SQLDialect`	Standard SQL (`1`)	No

Description

This option specifies whether the connector executes queries using standard SQL syntax or the legacy BigQuery SQL syntax.

- Standard SQL (`1`): The connector uses standard SQL.
- Legacy SQL (`0`): The connector uses legacy SQL.

Log Level

Key Name	Default Value	Required
`LogLevel`	OFF (`0`)	No

Description

Use this property to enable or disable logging in the connector and to specify the amount of detail included in log files.

! Important:

- Only enable logging long enough to capture an issue. Logging decreases performance and can consume a large quantity of disk space.
- When logging with connection strings and DSNs, this option only applies to per-connection logs.

Set the property to one of the following values:

- OFF (`0`): Disable all logging.
- FATAL (`1`): Logs severe error events that lead the connector to abort.
- ERROR (`2`): Logs error events that might allow the connector to continue running.
- WARNING (`3`): Logs events that might result in an error if action is not taken.
- INFO (`4`): Logs general information that describes the progress of the connector.
- DEBUG (`5`): Logs detailed information that is useful for debugging the connector.
- TRACE (`6`): Logs all connector activity.

When logging is enabled, the connector produces the following log files at the location you specify in the Log Path (`LogPath`) property:

- A `simbabigqueryodbcdriver.log` file that logs connector activity that is not specific to a connection.
- A `simbabigqueryodbcdriver_connection_[Number].log` file for each connection made to the database, where *[Number]* is a number that identifies each log file. This file logs connector activity that is specific to the connection.

Log Path

Key Name	Default Value	Required
`LogPath`	None	Yes, if logging is enabled.

Description

The full path to the folder where the connector saves log files when logging is enabled.

! Important:

When logging with connection strings and DSNs, this option only applies to per-connection logs.

Max File Size

Key Name	Default Value	Required
LogFileSize	20971520	No

Description

The maximum size of each log file in bytes. After the maximum file size is reached, the connector creates a new file and continues logging.

If this property is set using the Windows UI, the entered value is converted from megabytes (MB) to bytes before being set.

! Important:

When logging with connection strings and DSNs, this option only applies to per-connection logs.

Max Number Files

Key Name	Default Value	Required
LogFileCount	50	No

Description

The maximum number of log files to keep. After the maximum number of log files is reached, each time an additional file is created, the connector deletes the oldest log file.

! Important:

When logging with connection strings and DSNs, this option only applies to per-connection logs.

Minimum Query Results Size for HTAPI

Key Name	Default Value	Required
HTAPI_ MinResultsSize	1000	No

Description

When the number of table rows in your query results exceeds this number, and the number of pages in the results exceeds the **Ratio of Results to Rows per Block** (HTAPI_MinActivationRatio) value, the connector switches to using the BigQuery High-Throughput API instead of the REST API.

Minimum TLS

Key Name	Default Value	Required
Min_TLS	TLS 1.2 (1.2)	No

Description

The minimum version of TLS/SSL that the connector allows the data store to use for encrypting connections. For example, if TLS 1.1 is specified, TLS 1.0 cannot be used to encrypt connections.

- TLS 1.0 (1.0): The connection must use at least TLS 1.0.
- TLS 1.1 (1.1): The connection must use at least TLS 1.1.
- TLS 1.2 (1.2): The connection must use at least TLS 1.2.

OAuth Mechanism

Key Name	Default Value	Required
OAuthMechanism	User Authentication (1)	No

Description

The OAuth 2.0 authentication mechanism used to authenticate the connector.

- User Authentication (`1`): The connector authenticates as a user, through a Google user account.
- Service Authentication (`0`): The connector authenticates as a service, through a Google service account.

Path to CMEK

Key Name	Default Value	Required
`KMSKeyName`	None. The connector uses the default encryption key from Google.	No

Description

The resource ID of the customer-managed encryption key (CMEK) that you want the connector to use when executing queries. When this property is not set, the connector uses the default encryption key from Google.

For information about CMEKs and Cloud KMS encryption, see "Protecting Data with Cloud KMS Keys" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/customer-managed-encryption.

! Important:

- Do not set this property unless you are certain that you are specifying the correct CMEK. If you execute an INSERT statement with an incorrect CMEK, the connector returns an error or corrupts the table.
- When this property is set, the connector uses the specified CMEK for all queries.

Proxy Host

Key Name	Default Value	Required
`ProxyHost`	None	Yes, if connecting through a proxy server.

Description

The host name or IP address of a proxy server that you want to connect through.

Proxy Password

Key Name	Default Value	Required
ProxyPwd	None	Yes, if connecting to a proxy server that requires authentication.

Description

The password that you use to access the proxy server.

Proxy Port

Key Name	Default Value	Required
ProxyPort	None	Yes, if connecting through a proxy server.

Description

The number of the port that the proxy server uses to listen for client connections.

Proxy Username

Key Name	Default Value	Required
ProxyUid	None	Yes, if connecting to a proxy server that requires authentication.

Description

The user name that you use to access the proxy server.

Query Properties

Key Name	Default Value	Required
QueryProperties	None	No

Description

This option enables you to pass properties through to the server when inserting a job. Properties set in this manner are used for all queries in a connection. The `QueryProperties` list must be in the following form:

```
key1=value1,key2=value2,...,keyN=valueN
```

This option also supports the `time_zone` connection property in the following form:

```
time_zone=America/Los_Angeles
```

For detailed information, see "Supported Time Zone Values" in the Google BigQuery documentation: https://cloud.google.com/dataprep/docs/html/Supported-Time-Zone-Values_66194188.

! Important:

QueryProperties corresponds to the JobConfigurationQuery "connectionProperties" field. For detailed information, see "JobConfigurationQuery" in the Google BigQuery documentation: https://cloud.google.com/bigquery/docs/reference/rest/v2/Job#JobConfigurationQuery.

Ratio of Results to Rows Per Block

Key Name	Default Value	Required
`HTAPI_MinActivationRatio`	3	No

Description

The minimum ratio of total rows to rows in the first page needed to activate reading through the High-Throughput API.

If this value is set to 0, then the connector uses the High-Throughput API for all query results that meet the minimum results size specified by **Minimum Query Results Size for HTAPI** (`HTAPI_MinResultsSize`).

Refresh Token

Key Name	Default Value	Required
RefreshToken	None	Yes, if authenticating through a user account.

Description

The refresh token that you obtain from Google for authorizing access to BigQuery.

When you configure a DSN with the Windows connector, the refresh token is generated automatically after you provide the confirmation code.

When you configure a DSN with the Linux or macOS versions of the connector, you can use the Google OAuth 2.0 Playground to generate the token. For more information, see Using a Google User Account on page 36.

Request Google Drive Scope Access

Key Name	Default Value	Required
RequestGoogleDriveScope	Clear (0)	No

Description

This option specifies whether the connector requests access to Google Drive. Allowing the connector to access Google Drive enables support for federated tables that combine BigQuery data with data from Google Drive.

- Enabled (1): The connector requests access to Google Drive.
- Disabled (0): The connector does not request access to Google Drive.

Rows Fetched Per Block

Key Name	Default Value	Required
RowsFetchedPerBlock	100000	No

Description

The maximum number of rows that the connector can fetch for each data request.

Temporary Table Expiration Time

Key Name	Default Value	Required
LargeResultsTempTableExpirationTime	3600000	No

Description

The length of time, in milliseconds, for which any temporary tables exist. The minimum value is 3600000 milliseconds, or one hour.

Trusted Certificates

Key Name	Default Value	Required
TrustedCerts	The cacerts.pem file in the \lib subfolder within the connector's installation directory. The exact file path varies depending on the version of the connector that is installed. For example, the path for the Windows connector is different from the path for the macOS connector.	No

Description

The full path of the .pem file containing trusted CA certificates, for verifying the server.

If this option is not set, then the connector defaults to using the trusted CA certificates .pem file installed by the connector. To use the trusted CA certificates in the .pem file, set the UseSystemTrustStore property to 0 or clear the Use System Trust Store check box in the SSL Options dialog.

Use Default _bqodbc_temp_tables Large Results Dataset

Key Name	Default Value	Required
`UseDefaultLargeResultsDataset`	Clear (`0`)	No

Description

This option specifies whether the connector uses a default dataset to store temporary tables for large result sets if no dataset is specified and the data store specifies the US region.

- Enabled (`1`): The connector uses a default large result dataset of _bqodbc_temp_ tables.
- Disabled (`0`): The connector does not use a default large result dataset. If no dataset is specified, large result support is not enabled.

Note:

- If this value is specified, it overrides the `LargeResultsDatasetId` property.
- If this option is selected, the data store specifies the US region, and the _ bqodbc_temp_tables dataset does not exist, the connector creates this dataset.
- If you want to use large result sets with Legacy SQL, you must also enable the **Allow Large Result Sets** option. See Allow Large Result Sets on page 60.

Use Proxy Server

Key Name	Default Value	Required
N/A	Clear (`0`)	No

Description

This option specifies whether the connector uses a proxy server to connect to the data store.

- Enabled (`1`): The connector connects to a proxy server based on the information provided in the Proxy Host, Proxy Port, Proxy Username, and Proxy Password

fields or the `ProxyHost`, `ProxyPort`, `ProxyUID`, and `ProxyPWD` keys.

- Disabled (`0`): The connector connects directly to the BigQuery server.

Note:

This option is only available on Windows.

Use SQL_WVARCHAR instead of SQL_VARCHAR

Key Name	Default Value	Required
`UseWVarChar`	Clear (`0`)	No

Description

This option specifies how data types are mapped to SQL.

- Enabled (`1`): The connector returns data as SQL_WVARCHAR data instead of SQL_VARCHAR data.
- Disabled (`0`): The connector returns data as SQL_VARCHAR data.

Note:

This option applies only to result set columns that the connector would normally return as SQL_VARCHAR columns. It does not convert all columns into SQL_WVARCHAR.

Use System Trust Store

Key Name	Default Value	Required
`UseSystemTrustStore`	Clear (`0`)	No

Description

This option specifies whether to use a CA certificate from the system trust store, or from a specified `.pem` file.

- Enabled (`1`): The connector verifies the connection using a certificate in the system trust store.

- Disabled (`0`): The connector verifies the connection using a specified `.pem` file. For information about specifying a `.pem` file, see Trusted Certificates on page 73.

Note:

This option is only available on Windows.

Configuration Options Having Only Key Names

The following configuration options do not appear in the Windows user interface for the Simba Google BigQuery ODBC Connector. They are accessible only when you use a connection string or configure a connection on macOS or Linux.

Auth_Client_ID

Key Name	Default Value	Required
`Auth_Client_ID`	None	No

Description

The OAuth 2.0 client ID, which is used to generate the refresh token.

Important:

Only set this option if you are generating tokens based on your credentials.

Auth_Client_Secret

Key Name	Default Value	Required
`Auth_Client_Secret`	None	No

Description

The OAuth 2.0 client secret, which is used to generate the refresh token.

> **! Important:**
>
> Only set this option if you are generating tokens based on your credentials.

Driver

Key Name	Default Value	Required
`Driver`	`Simba Google BigQuery ODBC Connector` when installed on Windows, or the absolute path of the connector shared object file when installed on a non-Windows machine.	Yes

Description

On Windows, the name of the installed connector (`Simba Google BigQuery ODBC Connector`).

On other platforms, the name of the installed connector as specified in `odbcinst.ini`, or the absolute path of the connector shared object file.

FilterTablesOnDefaultDataset

Key Name	Default Value	Required
`FilterTablesOnDefaultDataset`	`FALSE`	No

Description

This option determines whether the connector filters tables in the `SQLTables` call and columns in the `SQLColumns` call to return only tables and columns that belong to the default dataset.

- `FALSE`: The connector returns all tables in the `SQLTables` call and all columns in the `SQLColumns` call.
- `TRUE`: The connector only returns tables and columns that belong to the default dataset.

Note:

To filter tables and columns, you must define a default dataset. For details, see Dataset on page 63.

When this option is set to `TRUE`, the connector behaves as described below for the functions SQLTables and SQLColumns.

For the function SQLTables:

Catalog	Schema	Table	Table Type	Returned List
NULL	NULL	NULL or %	NULL or %	All tables that belong to the default dataset under the default catalog
%	NULL	NULL or %	NULL or %	All tables that belong to the default dataset under all catalogs
NULL	%	NULL or %	NULL or %	All tables that belong to all schemas under the default catalog
%	%	NULL or %	NULL or %	All tables that belong to all schemas under all catalogs
NULL	<schema>	NULL or %	NULL or %	All tables that belong to the specified schema under the default catalog

Catalog	Schema	Table	Table Type	Returned List
<catalog>	<schema>	NULL or %	NULL or %	All tables that belong to the specified schema under the specified catalog

For the function SQLColumns:

Catalog	Schema	Table	Column	Returned List
NULL	NULL	NULL	NULL	All columns of all tables that belong to the default dataset under the default catalog
<catalog>	NULL	NULL	NULL	All columns of all tables that belong to the default dataset under the speficied catalog
NULL	%	NULL	NULL	All columns of all tables that belong to all datasets under the default catalog
<catalog>	%	NULL	NULL	All columns of all tables that belong to all datasets under the specified catalog
NULL	<schema>	NULL	NULL	All columns of all tables that belong to the specified dataset under the default catalog
<catalog>	<schema>	NULL	NULL	All columns of all tables that belong to the specified dataset under the specified catalog

IgnoreTransactions

Key Name	Default Value	Required
`IgnoreTransactions`	`0`	No

Description

This option determines whether the connector ignores attempts to perform transactions.

- `0`: Attempts to perform transactions produce a user alert.
- `1`: The connector ignores attempts to perform transactions. No alerts are generated for these calls.

P12CustomPwd

Key Name	Default Value	Required
`P12CustomPwd`	None	No

Description

This option allows you to use a P12 private key file for SSL encryption. The specified value is the password your key file is encrypted with.

Timeout

Key Name	Default Value	Required
`Timeout`	`300`	No

Description

The length of time, in seconds, for which the connector retries a failed API call before timing out. The specified value must be an integer. A value of 0 indicates no timeout.

UseQueryCache

Key Name	Default Value	Required
`UseQueryCache`	`1`	No

Description

This option determines whether the connector uses the query cache when retrieving results.

- `1`: The connector uses cached query results, if they are available.
- `0`: The connector does not use the query cache.

For detailed information about cached query results, see "Using Cached Query Results" in the Google Cloud Platform documentation: https://cloud.google.com/bigquery/docs/cached-results.

Third-Party Trademarks

Debian is a trademark or registered trademark of Software in the Public Interest, Inc. or its subsidiaries in Canada, United States and/or other countries.

Linux is the registered trademark of Linus Torvalds in Canada, United States and/or other countries.

Mac, macOS, Mac OS, and OS X are trademarks or registered trademarks of Apple, Inc. or its subsidiaries in Canada, United States and/or other countries.

Microsoft, MSDN, Windows, Windows Server, Windows Vista, and the Windows start button are trademarks or registered trademarks of Microsoft Corporation or its subsidiaries in Canada, United States and/or other countries.

Red Hat, Red Hat Enterprise Linux, and CentOS are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in Canada, United States and/or other countries.

SUSE is a trademark or registered trademark of SUSE LLC or its subsidiaries in Canada, United States and/or other countries.

Ubuntu is a trademark or registered trademark of Canonical Ltd. or its subsidiaries in Canada, United States and/or other countries.

Google BigQuery, Google, and BigQuery are trademarks or registered trademarks of Google, Inc. or its subsidiaries in Canada, the United States and/or other countries.

All other trademarks are trademarks of their respective owners.