Progress DataDirect for ODBC for Google BigQuery

User's Guide

Release 8.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Google BigQuery Driver

The Progress® DataDirect® for ODBC™ for Google BigQuery™ driver supports the standard and legacy SQL dialects of Google BigQuery. It allows create, read, update, and delete operations in internal tables (stored inside Google BigQuery) and read operations in external tables (stored in data sources outside Google BigQuery). It leverages the SQL engine functionality of Google BigQuery to execute SQL queries.

The documentation for the driver also includes the *Progress DataDirect for ODBC Drivers Reference*. The reference provides general reference information for all DataDirect drivers for ODBC, including content on troubleshooting, supported SQL escapes, and DataDirect tools. For the complete documentation set, visit to the Progress DataDirect Connectors Documentation Hub: https://docs.progress.com/bundle/datadirect-connectors/page/DataDirect-Connectors-by-data-source.html.

For details, see the following topics:

- Contacting Technical Support

What's new in this release?

Support and Certifications

Visit the following web pages for the latest support and certification information.

- Release Notes: https://www.progress.com/odbc/release-history/
- DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf

Changes Since 8.0.0

- **Enhancements**
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The driver has been enhanced to retry API call executions when an HTTP failure or driver exception occurs. You can configure this behavior with the new Retry Exceptions (RetryExceptions) connection option. When this option is set to 1, the driver uses the retry value specified by the WS Retry Count (WSRetryCount) connection option. See Retry Exceptions on page 108 for details.
  - The driver has been enhanced to support fetching access and refresh tokens at connection when OAuth2.0 is enabled. When using the new dynamic authorization code grant, you can initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or a third-party application. In addition, the new Enable Login Prompt (EnableLoginPrompt) option has been added to enable this functionality. See Dynamic authorization code grant on page 51 and Enable Login Prompt on page 87 for details.
  - The driver has been enhanced to support the JSON and Interval data types, which map to the SQL_LONGVARCHAR and SQL_VARCHAR odbc data types respectively. See Data types on page 21 for details.
  - On Windows platforms, the driver includes an enhanced setup dialog, the Progress DataDirect Google BigQuery Configuration Manager, for quick configuration and testing of your driver in a web browser. The tool allows you to:
    - Configure data sources
    - Generate and edit connection strings
    - Test connect data sources and connection strings
    - Execute SQL commands for testing
    - Access connection option descriptions and the full product documentation

    For details, see Generating connection strings with the Configuration Manager on page 46 and Testing connections and queries with the Configuration Manager on page 47.
  - The driver has been enhanced to allow users to specify values for the following exposed connection options:
    - Auth URI and Token URI when connecting with OAuth 2.0 authentication.

- JWT Audience and Token URI when connecting with Service Account authentication.

  The driver used the default values for these options previously when authenticating to Google BigQuery. See Configuring OAuth 2.0 authentication on page 50 and Configuring service account authentication on page 53 for details.

- The driver has been enhanced to support the Google BigQuery Streaming API when executing batch insert operations. This behavior can be configured using the Use Streaming Insert connection option. See Google BigQuery Streaming API for details.
- A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and odbc.ini files. At connection, the driver decrypts these passwords and passes them to the data source as required. See Password Encryption Tool (UNIX/Linux only) on page 48 for details.
- The driver has been enhanced with the new Primary Key Pattern (PrimaryKeyPattern) connection option, which allows you to determine which column in a table is designated as the primary key. Google BigQuery does not have the concept of primary keys, or even uniqueness. However, some applications do not function properly without at least one column in a table designated as the primary key. This option allows your applications that require a primary key to function correctly when connecting to Google BigQuery data sources.
- The driver has been enhanced to support the BIGNUMERIC data type. See Data types on page 21 for more information.
- The Service Account Key Content connection option has been added to the driver. This option allows you to specify the private key required for service account authentication without having to persist the `.json` or `.p12` file that contains the private key. See Configuring service account authentication on page 53 and Service Account Key Content on page 112 for details.
- The Enable Catalog Support connection option has been added to the driver. It determines whether the driver supports specifying values for catalog parameters in metadata calls. See Enable Catalog Support on page 87 for details.
- The drivers using base version B0649 and later have been enhanced to include timestamp in the internal packet logs by default. If you want to disable the timestamp logging in packet logs, set `PacketLoggingOptions=1`. The internal packet logging is not enabled by default. To enable it, set `EnablePacketLogging=1`.
- The driver has been enhanced to support the Google BigQuery Storage API when fetching large result sets. See Google BigQuery Storage API and Storage API attributes for details.

  Note: Currently, the Storage API is supported only on Windows 64-bit, Linux 64-bit, and JVM 64-bit. If an application attempts to use the Storage API on an unsupported platform, the driver falls back to the Standard API.

- The Driver Manager for UNIX/Linux has been enhanced to support setting the Unicode encoding type for applications on a per connection basis. By passing a value for the SQL_ATTR_APP_UNICODE_TYPE attribute using SQLSetConnectAttr, your application can specify the encoding at connection. This allows your application to pass both UTF-8 and UTF-16 encoded strings with a single environment handle.

  The valid values for the SQL_ATTR_APP_UNICODE_TYPE attribute are SQL_DD_CP_UTF8 and SQL_DD_CP_UTF16. The default value is SQL_DD_CP_UTF8.

  This enhancement is available in build 08.02.0449 of the driver manager.

- **Changed Behavior**
  - On Windows platform, the default value of the SQL Engine Mode connection option has been changed to 1 (Server).

- When existing DSNs are updated using the Configuration Manager, the boolean values of "0" and "1" are converted to "true" and "false" for relevant connection options.
- The driver updates string values in an existing DSN to number values when the Configuration Manager is used to update the Authentication Method connection option. The following updates occur:

  - The string value "oauth2.0" is updated to 24

  - The string value "serviceaccount" is updated to 28
- The default value of the WSRetryCount connection option has been changed to 5.
- The default value of the SchemaSet configuration option has been changed to the project and dataset specified at connection. For details, see SchemaSet (Config Option) on page 82

Highlights of 8.0.0 Release

- The driver supports both standard and legacy Google BigQuery SQL dialects. See Standard and legacy SQL support on page 22 for more information.
- The driver serves as a complete pass-through driver. It leverages Google BigQuery SQL engine to execute queries.
- The driver supports create, read, update, and delete (CRUD) operations.
- The driver returns data for complex data types, such as Array and Struct, as JSON strings, which are easy to comprehend for ODBC applications. See Json Format on page 91 for more information.
- The driver provides proxy support. See Proxy Host on page 103, Proxy Password on page 103, Proxy Port on page 104, and Proxy User on page 104 for more information.
- The driver supports the handling of large result sets with configurable paging and the Fetch Size on page 88 and WS Fetch Size on page 119 connection options.

Driver requirements

Data source and platform requirements

For the latest support information, visit the DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

Java requirements

- The driver requires a Java Virtual Machine (JVM) that is Java SE 8 or higher. JVM support includes Oracle JDK, OpenJDK, and IBM SDK (Java) distributions.
- For 32-bit drivers, a 32-bit Java Virtual Machine (JVM) is required. For 64-bit drivers, a 64-bit Java Virtual Machine (JVM) is required.
- For Windows, you must set the PATH environment variable to the directory containing the `jvm.dll` for your JVM.
- For Linux, you must set the library path environment variable of your operating system to the directory containing your JVM's `libjvm.so` file and that directory's parent directory.

Windows requirements for 32-bit drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- You must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Windows requirements for 64-bit drivers

- All required network software that is supplied by your database system vendors must be 64-bit compliant.
- You must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Linux requirements for 32-bit drivers

- If your application was built with 32-bit system libraries, you must use 32-bit drivers. The database to which you are connecting can be either 32-bit or 64-bit enabled.
- An application compatible with components that were built using g++ GNU project C++ Compiler version 3.4.6 and the Linux native pthread threading model (Linuxthreads).

Linux requirements for 64-bit drivers

- An application compatible with components that were built using g++ GNU project C++ Compiler version 3.4 and the Linux native pthread threading model (Linuxthreads).

AIX requirements for 32-bit and 64-bit drivers

- IBM POWER processor
- An application compatible with components that were built using Visual Age C++ 6.0.0.0 and the AIX native threading model.

HP-UX requirements for 32-bit drivers

- Intel Itanium II (IPF) processor is supported.
- An application compatible with components that were built using HP aC++ 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads).

Note:

- Do not link with the –lc linker option.
- Set the LD_PRELOAD environment variable to the `libjvm.so` from your JVM installation.

HP-UX requirements for 64-bit drivers

- HP aC++ v. 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads)

Note:

- Do not link with the –lc linker option.
- Set the `LD_PRELOAD` environment variable to the `libjvm.so` of your JVM installation.

Oracle Solaris requirements for 32-bit drivers

- x64: Intel and AMD processor is supported.
- An application compatible with components that were built using Oracle C++ 5.8 and the Solaris native (kernel) threading model

Oracle Solaris requirements for 64-bit drivers

- The following processors are supported:
  - Oracle SPARC
  - x64: Intel and AMD
- For Oracle SPARC: An application compatible with components that were built using Sun Studio 11, C++ compiler version 5.8 and the Solaris native (kernel) threading model
- For x64: An application compatible with components that were built using Sun C++ Compiler version 5.8 and the Solaris native (kernel) threading model

Installing and setting up the driver (Windows)

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:
   a) After downloading the product, unzip the installer files to a temporary directory.
   b) From the installer directory, run the appropriate installer file to start the installer. The installer file takes the following form:

      `PROGRESS_DATADIRECT_ODBC_nn_WIN_xx_INSTALL.exe`

   c) Follow the prompts to complete installation.

   Note:

   The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for ODBC Drivers Installation Guide*.

2. Before you can use your driver, you must set the PATH environment variable to include the path of the `jvm.dll` file of your Java™ Virtual Machine (JVM).
3. To configure the driver using the ODBC Administrator (GUI), start the ODBC Administrator from the Progress DataDirect program group. The GUI dialog allows you to configure the data source definitions in the Windows Registry or generate connection strings.

   Note: The Windows driver also supports using connection strings to connect to your service. For more information, see Using a connection string on page 47.

4. Select either the **User DSN** or **System DSN** tab to display a list of data sources.

- **User DSN**: If you installed a default DataDirect ODBC user data source as part of the installation, select the appropriate data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.

- **System DSN**: To configure a new system data source, click **Add** to display a list of installed drivers. Select the your driver and click **Finish** to display the driver Setup dialog box.

5. The **Connection** tab of the Configuration Manager opens in a window. Provide values for the following essential connection options; then, click **Apply**:

   For all connections:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Project**: The name of the project that you want the driver to connect to. The projects in Google BigQuery are equivalent to catalogs in ODBC.
   - **Dataset**: The name of the dataset that you want the driver to connect to. The datasets in Google BigQuery are equivalent to schemas in ODBC.

   For OAuth 2.0 refresh token grant:

   - **Authentication Method**: Determines which authentication method the driver uses during the course of a session. Set to **OAuth2**. This is the default authentication method.
   - **Refresh Token**: Enter the refresh token used to either request a new access token or renew an expired one.

     Important: The Refresh Token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

   - **Client ID**: Enter the consumer key to connect to your Google BigQuery instance.
   - **Client Secret**: Enter the client secret to connect to your Google BigQuery instance.

   For OAuth 2.0 access token flow:

   - **Authentication Method**: Determines which authentication method the driver uses during the course of a session. Set to **OAuth2**. This is the default authentication method.
   - **Access Token**: Enter the access token required to authenticate to your Google BigQuery instance.

     Important: The Access Token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

   - **Client ID**: Enter the consumer key to connect to your Google BigQuery instance.
   - **Client Secret**: Enter the client secret to connect to your Google BigQuery instance.

   Note: See Configuring OAuth 2.0 authentication on page 50 for details

6. Set the values for any additional connection options that you want to configure. To view more options, click on the tabs on the dialog. See the following resources for additional information on optional features and functionality:

   - Connection option descriptions on page 69 provides a complete list of supported options by functionality.
   - Performance considerations on page 57 describes connection options that affect performance, along with recommended settings.
   - Configuring data sources with the Configuration Manager guides you through using the GUI to configure the driver.

   Note: For most connections, specifying the minimum required connection options is sufficient to begin accessing data; however, you can provide values for optional connection options to use additional supported features and improve performance.

7. Click **Test Connect** to attempt to connect to the data source using the connection options.
8. If you are using OAuth 2.0 authentication, the logon dialog appears. Update the following fields; then, click **OK**.

   - **Client ID**: Enter the consumer key that is used to connect to the instance.
   - **Client Secret**: Enter the client secret that is used to connect to the instance.

     Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

   - **Access Token**: Enter the access token required to authenticate to the instance.
   - **Refresh Token**: Enter the refresh token used to either request a new access token or renew an expired one.
   - **Scope**: Enter an OAuth scope or a space-separated list of OAuth scopes that limit the permissions granted by an access token.

   Note: The information you enter in the logon dialog box during a test connect is not saved.

9. If the test was successful, the window displays a confirmation message.
10. Click **OK** to close the setup dialog. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the Configure Manager to modify your data source, or you can override these defaults by connecting to the data source using a connection string with alternate values.
11. Connect to your Google BigQuery instance and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:

    - Example application: The example application allows you to test connect, execute SQL statements, and practice using the ODBC API right out of the box.
    - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
    - Microsoft Excel: Excel is a spreadsheet tool that allows you to connect, view tables, and execute SQL statements against your data.

This completes the deployment of the driver.

Installing and setting up the driver (UNIX/Linux)

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:
   a) After downloading the product, extract the contents of the product file.
   b) From the installer directory, run the installer's binary file to start the installer. The file for the installer program takes the following form:

      `PROGRESS_DATADIRECT_ODBC_nn_LINUX_xx_INSTALL.bin`

   c) Follow the prompts to complete installation.

   The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for ODBC Drivers Installation Guide*.

2. Configure the environment variables:
   a) Check your permissions: You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.
   b) Run one of the following product setup scripts from the installation directory to set variables: `odbc.sh` or `odbc.csh`. For Korn, Bourne, and equivalent shells, execute `odbc.sh`. For a C shell, execute `odbc.csh`. Executing the setup script:
      - Sets the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For details, see ODBCINI on page 37.
      - Sets the library path environment variable for your operating system, `LD_LIBRARY_PATH`, to include the directory containing your JVM's `libjvm.so` file. For details, see Library search path on page 36.

   c) Set the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For example, if you use an installation directory of `/opt/odbc` and the default system information file name, you would enter:
      - **Korn or Bourne shell**: `ODBCINI=/opt/odbc/odbc.ini; export ODBCINI`
      - **C shell**: `setenv ODBCINI /opt/odbc/odbc.ini`

   d) Set the library path environment variable for your Linux operating system to include the directory containing your JVM's `libjvm.so` file, and that directory's parent directory. The Library Path Environment Variable on Linux is `LD_LIBRARY_PATH`.

3. Configure the driver using one of the following methods:
   - **odbc.ini file**: You can begin using the driver immediately by editing the `odbc.ini` file in the installation directory with a text editor. The following demonstrates a data source definition with the minimal options.

     ```
     [ODBC Data Sources]
     Google BigQuery=DataDirect 8.0 Google BigQuery

     [Google BigQuery]
     Driver=ODBCHOME/lib/xxgbq28.yy
     Project=myproject
     ```

```
Dataset=mydataset
AccessToken=abcdefghi12345678
RefreshToken=wxyz123456789
ClientID=123abc.apps.googleusercontent.com
ClientSecret=ab123xy
```

See Configuration through the system information (odbc.ini) file on page 39 for more information.

Note: The User and Password options are not required to be stored in the data source. They can also be sent separately by the application using the SQLConnect ODBC API. For SQLDriverConnect and SQLBrowseConnect, they will need to be specified in the data source or connection string.

- **OAuth 2.0 access token flow**

```
[ODBC Data Sources]
Google BigQuery=DataDirect 8.0 Google BigQuery


[Google BigQuery]
Driver=ODBCHOME/lib/xxgooglebigquery28.yy
...
ClientID=client_id
...
ClientSecret=client_secret
...
AccessToken=access_token
...
```

See Configuration through the system information (odbc.ini) file on page 39 for more information.

- **Connection string**: The driver also supports using connection strings for DSN (data source name), File DSN, or DSN-less connections. See Using a connection string on page 47 and DSN-less connectionsfor more information.

Note: For most connections, specifying the minimum required connection options is sufficient to begin accessing data; however, you can provide values for optional connection options to use additional supported features and improve performance.

4. Set the values for any additional options that you want to configure. For additional information on optional features and functionality, see the following resources:
   - Connection option descriptions on page 69 provides a complete list of supported options by functionality.
   - Performance considerations on page 57 describes connection options that affect performance, along with recommended settings.
5. Connect to your Google BigQuery instance and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:
   - Example application: The example application allows you to test connect, execute SQL statements, and practice using the ODBC API right out of the box.
   - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
   - Microsoft Excel: Excel is a spreadsheet tool that allows you to connect, view tables, and execute SQL statements against your data.

This completes the deployment of the driver.

ODBC compliance

The Progress DataDirect for ODBC for Google BigQuery driver is compliant with the Open Database Connectivity (ODBC) specification.

The Google BigQuery driver supports only the following Level 2 functions:

- SQLColumnPrivileges
- SQLDescribeParam
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLTablePrivileges

Version string information

The driver for Google BigQuery has a version string of the format:

XX.YY.ZZZZ(B*AAAA*, U*BBBB*, J*DDDDDD*)

The Driver Manager on UNIX and Linux has a version string of the format:

XX.YY.ZZZZ(U*BBBB*)

The component for the Unicode conversion tables (ICU) has a version string of the format:

XX.YY.ZZZZ

where:

XX is the major version of the product.

YY is the minor version of the product.

ZZZZ is the build number of the driver or ICU component.

AAAA is the build number of the driver's bas component.

BBBB is the build number of the driver's utl component.

DDDDDD is the version of the Java components used by the driver.

For example:

```
08.00.0017 (B0254, U0180, J000109)
        |__|  |___|  |___|   |____|
       Driver  Bas    Utl     Java
```

 On Windows, you can check the version string through the properties of the driver DLL. Right-click the driver DLL and select **Properties**. The Properties dialog box appears. On the Details tab, the **File Version** will be listed with the other file properties.

You can always check the version string of a driver on Windows by looking at the About tab of the driver's Setup dialog.

UNIX® On UNIX and Linux, you can check the version string by using the test loading tool shipped with the product. This tool, ivtestlib for 32-bit drivers and ddtestlib for 64-bit drivers, is located in *install_directory*/bin.

The syntax for the tool is:

```
ivtestlib shared_object
```

or

```
ddtestlib shared_object
```

For example, for the 32-bit driver on Linux:

```
ivtestlib ivgbq28.so
```

returns:

```
08.00.0001 (B0002, U0001, J000003)
```

For example, for the Driver Manager on Linux:

```
ivtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

For example, for the 64-bit Driver Manager on Linux:

```
ddtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

For example, for the 32-bit ICU component on Linux:

```
ivtestlib libivicu28.so
08.00.0001
```

Note: On AIX, Linux, and Solaris, the full path to the driver does not have to be specified for the test loading tool. The HP-UX version of the tool, however, requires the full path.

---

getFileVersionString function

Version string information can also be obtained programmatically through the function `getFileVersionString`. This function can be used when the application is not directly calling ODBC functions.

This function is defined as follows and is located in the driver's shared object:

```
const unsigned char* getFileVersionString();
```

This function is prototyped in the `qesqlext.h` file shipped with the product.

Data types

The following table lists the data types supported by the Google BigQuery driver, and how they map to the ODBC data types.

Table 1: Google BigQuery data types

Google BigQuery Data Type	ODBC Data Type
STRING	SQL_WVARCHAR
ARRAY	SQL_WVARCHAR
RECORD	SQL_WVARCHAR
GEOGRAPHY	SQL_WVARCHAR
BOOL	SQL_BIT
BYTES	SQL_VARBINARY
INTERVAL	SQL_VARCHAR
INT64	SQL_BIGINT
JSON	SQL_LONGVARCHAR
NUMERIC	SQL_DECIMAL
BIGNUMERIC	SQL_DECIMAL
FLOAT64	SQL_DOUBLE
DATE	SQL_TYPE_DATE
TIME	SQL_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP
DATETIME	SQL_TYPE_TIMESTAMP

Standard and legacy SQL support

The driver supports both standard and legacy SQL dialects of Google BigQuery. By default, the driver uses standard SQL to execute queries. However, you can change the dialect to legacy SQL either by adding the prefix `#legacySQL` to a query or by setting the Syntax connection option to `legacy` (`Syntax=legacy`).

The following example demonstrates how to use the prefix `#legacySQL` in a query:

```
#legacySQL
SELECT ID,name FROM [bigquery-public-data:samples.EMP]
WHERE name CONTAINS "RA";
```

Note: In both standard and legacy SQL dialects, some of the identifiers, the names associated with SQL objects, are case-sensitive. For more information about identifiers and the complete list of case-sensitive identifiers, refer to the Google BigQuery documentation.

See also

Syntax on page 116
Using identifiers on page 22

Using identifiers

Identifiers are used to refer to objects exposed by the driver, such as tables, columns, or caches. Certain identifiers are case-sensitive, as defined by Google BigQuery. The driver supports both unquoted and quoted identifiers for naming objects.

An unquoted identifier must start with an ASCII alpha character and can be followed by zero or more ASCII alpha or numeric characters. Quoted identifiers must be enclosed in backticks (``). A quoted identifier can contain any Unicode character, including the space character. The Google BigQuery driver recognizes the Unicode escape sequence \uxxxx as a Unicode character.

The maximum length of both quoted and unquoted identifiers is 1024 characters.

Note: For more information about identifiers and the complete list of case-sensitive identifiers, refer to the Google BigQuery documentation.

Troubleshooting

The *Progress DataDirect for ODBC Drivers Reference* provides information on troubleshooting problems should they occur.

Refer to the "Troubleshooting" section in the *Progress DataDirect for ODBC Drivers Reference* for details.

Contacting Technical Support

Progress DataDirect offers a variety of options to meet your support needs. Please visit our Web site for more details and for contact information:

https://www.progress.com/support

The Progress DataDirect Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

When you contact us for assistance, please provide the following information:

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.
- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

August 2019, Release 8.0.0 for the Progress DataDirect for ODBC for Google BigQuery Driver, Version 0001

2

Tutorials

The following sections guide you through using the driver to access your data with some common third-party applications:

- Accessing data in Tableau (Windows only) on page 27
- Accessing data in Microsoft Excel (Windows only) on page 29
- Accessing data in Microsoft Excel from the Query Wizard (Windows only) on page 31

For details, see the following topics:

- The Example application
- Accessing data in Tableau (Windows only)
- Accessing data in Microsoft Excel (Windows only)
- Accessing data in Microsoft Excel from the Query Wizard (Windows only)

The Example application

The driver installation includes an ODBC application called Example that can be used for:

- Testing any type of SQL statement
- Testing database connections
- Verifying your database environment

It can also be used to demonstrate ODBC function calls, including the following:

- SQLAllocHandle
- SQLBindCol
- SQLBindParameter
- SQLColAttribute
- SQLConnect
- SQLDescribeCol
- SQLDescribeParam
- SQLDisconnect
- SQLDriverConnect
- SQLExecDirect
- SQLFetch
- SQLFreeHandle
- SQLFreeStmt
- SQLGetDiagRec
- SQLGetInfo
- SQLNumResultCols
- SQLPrepare
- SQLSetEnvAttr
- SQLSetStmtAttr

The Example application can be built using the files located in the `\samples\examples` directory of the DataDirect for ODBC Drivers installation directory.

Note:

- For Windows, you can build the Windows app for ANSI and Unicode.
- For UNIX/Linux, instructions for building the Example application are contained inside the file `example.mak`, which can be read with a text editor.

To use the Example application:

1. After you have configured the data source, navigate to the `instal_dir\samples\example` directory.
2. Open the application using one of the following methods:
   - Running the application executable or binary:
     - On Windows, double-click the `Example.exe` file.
     - On UNIX/Linux, run the `example` application.
   - Executing a command-line argument. For example:
     - `example` *`connection_string`*
     - `example "DSN" "UID" "PWD"`
     - `example` *`connection_string`* `"`*`sql_command_1`*`" ["`*`sql_command_2`*`" ...]`

   Results: A command prompt opens.
3. Follow the prompts to enter your data source name, user name, and password. If successful, a SQL> prompt appears.
4. At the prompt, enter SQL statements to test your connection. For example:

   ```
   SELECT * FROM INFORMATION_SCHEMA.TABLES
   ```

The results of your query are displayed. If example is unable to connect, the appropriate error message is returned.

Accessing data in Tableau (Windows only)

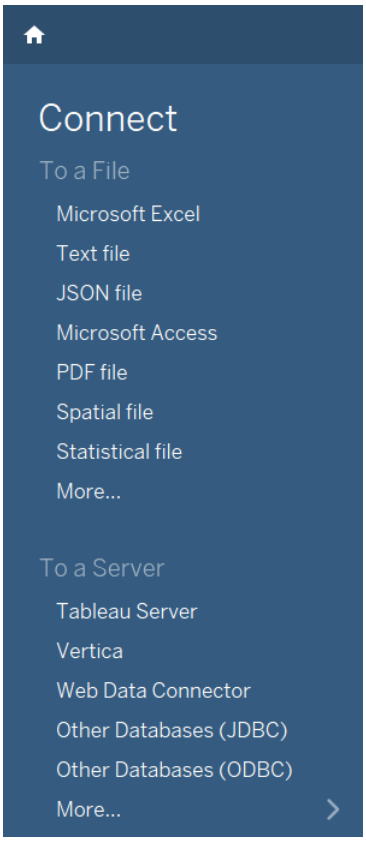
After you have configured your data source, you can use the driver to access your Google BigQuery data with Tableau. Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Tableau, you can improve performance when retrieving data while leveraging the driver's relational mapping tools.

To use the driver to access data with Tableau:

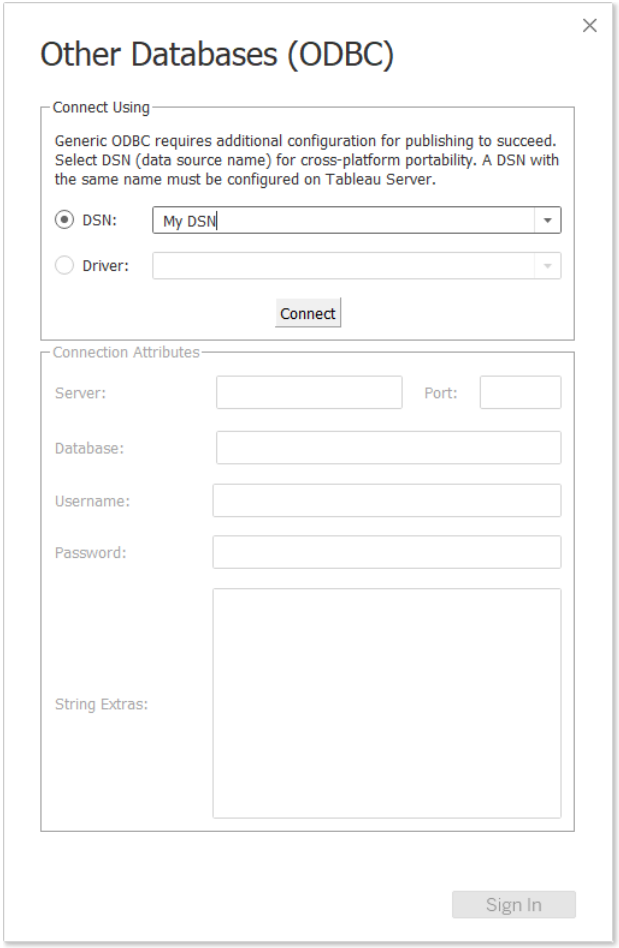
1. Navigate to the `\tools\Tableau` subdirectory of the Progress DataDirect installation directory; then, locate the Tableau data source file, `DataDirect GoogleBigQuery.tdc`.
2. Copy the `DataDirect GoogleBigQuery.tdc` into the following directory:

   `C:\Users\`*`user_name`*`\Documents\My Tableau Repository\Datasources`

3. Open Tableau. If the **Connect** menu does not open by default, select **Data > New Data Source** or the Add New Data Source button to open the menu.

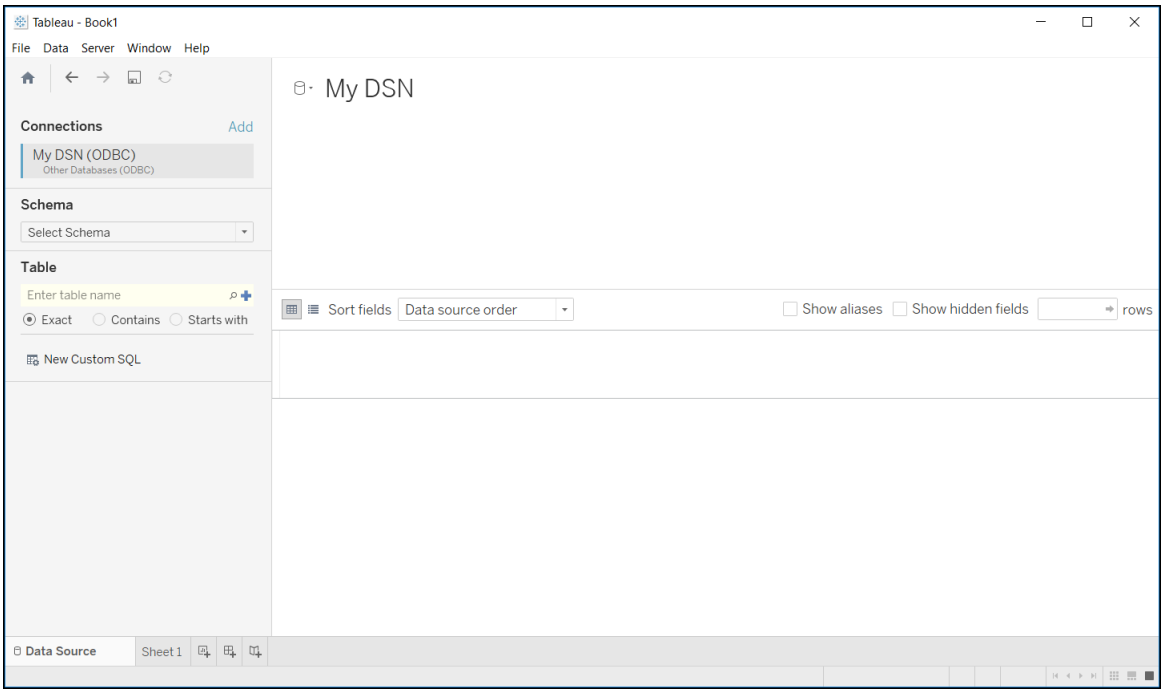


4. From the **Connect** menu, select **Other Databases (ODBC)**.
5. The **Server Connection** dialog appears.



In the DSN field, select the data source you want to use from the drop-down menu. For example, **My DSN**. Then, click **Connect**.

6. Click **OK** on the Server Connection dialog.
7. The **Data Source** window appears.



By default, Tableau connects live, or directly, to your data. We recommend that you use the default settings to avoid extracting all of your data. However, if you prefer, you can import your data by selecting the **Extract** option at the top of the dialog.

8. In the Schema field, select the database you want to use. The tables stored in this database are now available for selection in the Table field.

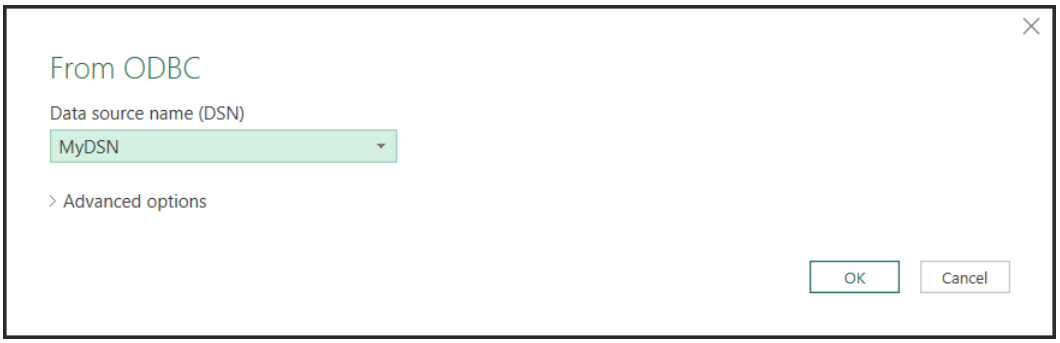
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

Accessing data in Microsoft Excel (Windows only)

After you have configured your data source, you can use the driver to access your data with Microsoft Excel from the Data Connection Wizard. Using the driver with Excel provides improved performance when retrieving data, while leveraging the driver's relational-mapping tools.

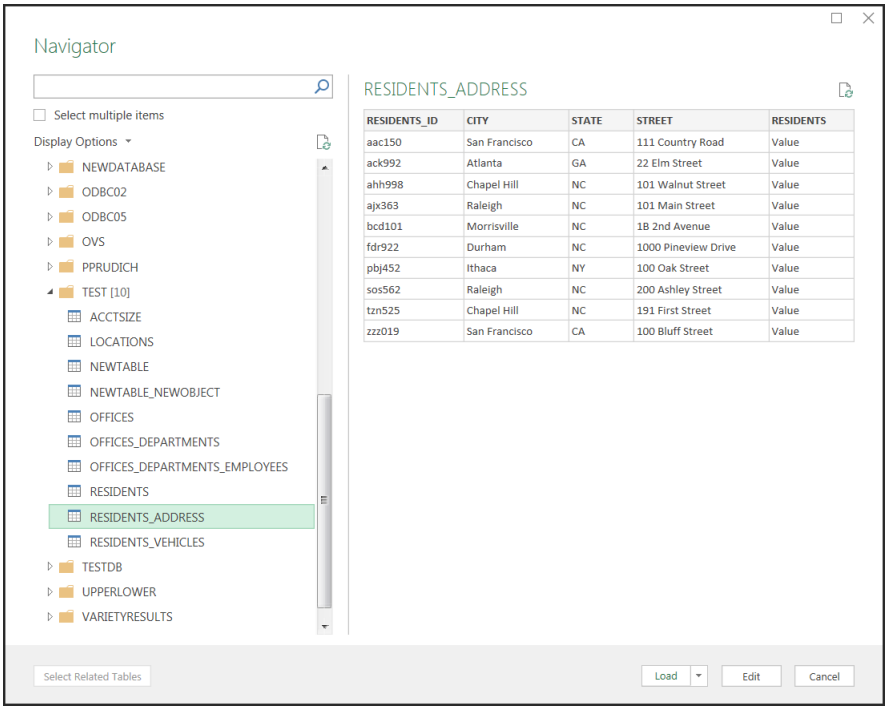
To use the driver to access data with Excel from the Data Connection Wizard:

1. Open your workbook in Excel.
2. From the **Data** menu, select **Get Data**>**From Other Sources**>**From ODBC**.
3. The **From ODBC** dialog appears.



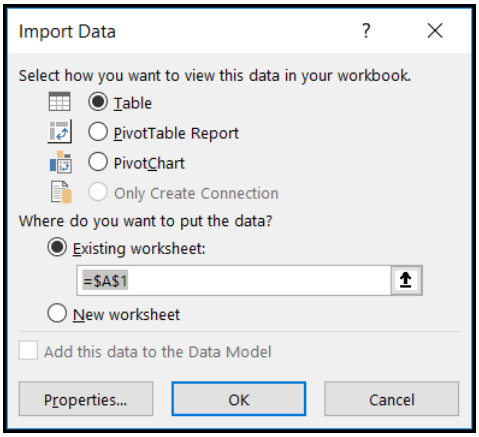
Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

4. You are prompted for logon credentials for your data source:
   - If your data source does not require logon credentials or if you prefer to specify your credentials using a connection string, select **Default or Custom** from the menu on the left. Optionally, specify your credential-related properties using a connection string in the provided field. Click **Connect** to proceed.
   - If your data source uses Windows credentials, select **Windows** from the menu; then, provide your credentials. Optionally, specify a connection string with credential-related properties in the provided field. Click **Connect** to proceed.
   - If your data source uses credentials stored on the database, select **Database**; then, provide your user name and password. Optionally, specify a connection string in the provided field. Click **Connect** to proceed.
5. The **Navigator** window appears.



From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

6. Load your data:
   - Click **Load** to import your data into your work sheet. Skip to the end.
   - Click **Load**>**Load To** to specify a location to import your data. Proceed to the next step.
7. The **Import Data** window appears.



   Select the desired view and insertion point for the data. Click **OK**.

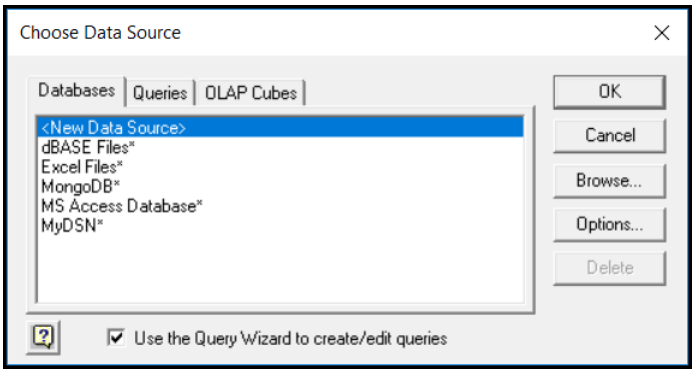
You have successfully accessed your data in Excel. For more information, refer to the Microsoft Excel product documentation at: https://support.office.com/.

Accessing data in Microsoft Excel from the Query Wizard (Windows only)

After you have configured your data source, you can use the driver to access your data with Microsoft Excel from the Query Wizard. Using the driver with Excel provides improved performance when retrieving data, while leveraging the driver's relational-mapping tools.

To use the driver to access data with Excel from the Query Wizard:

1. Open your workbook in Excel.
2. From the **Data** menu, select **Get Data**>**From Other Sources**>**From Microsoft Query**.
3. The **Choose Data Source** dialog appears.



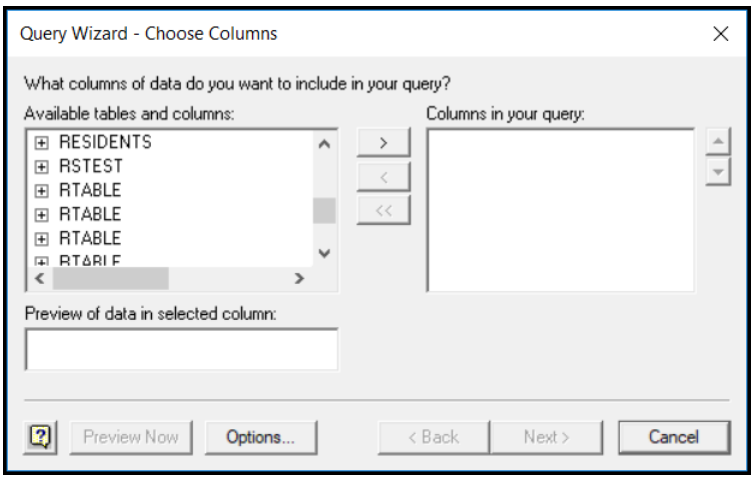
From the Databases list, select your data source. For example, **MyDSN**. Click **OK**.

4. The logon dialog appears pre-populated with the connection information you provided in your data source. If required, type your password. Click **OK** to proceed.

Note: The logon dialog may reappear if Excel needs to access additional information from the data source. If this occurs, re-enter your password; then, click **OK** to proceed to the next step.

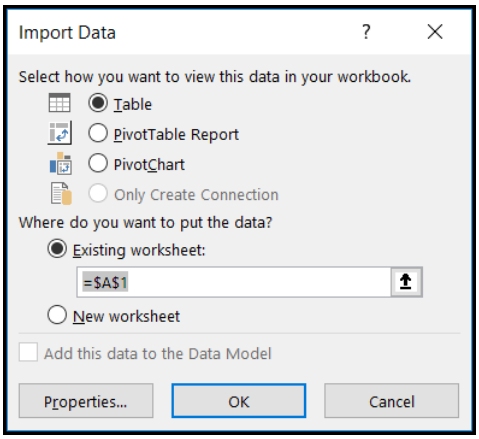
---

5. The **Query Wizard - Choose Columns** window appears.



Choose the columns you want to import into your workbook. To add a column, select the column name in Available tables and columns pane; then, click the **>** button. After you add the columns you want to include, click **Next** to continue.

6. Optionally, filter your data using the drop-down menus; then, click **Next**.
7. Optionally, sort your data using the drop-down menus; then, click **Next**.
8. Select "Return Data to Microsoft Excel"; then, click **Finish**.
9. The **Import Data** window appears.



Select the desired view and insertion point for your data. Click **OK**.

You have successfully accessed your data in Excel using the Query Wizard. For more information, refer to the Microsoft Excel product documentation at: https://support.office.com/.

3

Using the driver

This section guides you through the configuring and connecting to data sources. In addition, it explains how to use the functionality supported by your driver.

For details, see the following topics:

- Configuring and connecting to data sources
- Connecting through a proxy server
- Using security
- Performance considerations
- Internal and external tables support
- Google BigQuery Storage API
- Google BigQuery Streaming API
- DataDirect connection pooling
- Timeouts
- Isolation and lock levels supported
- Unicode support
- Parameter metadata support

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. Information that the driver needs to connect to a database is stored in a *data source*. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on UNIX/Linux), and file data sources. On Windows, user and system data sources are stored in the registry of the local computer. The difference is that only a specific user can access user data sources, whereas any user of the machine can access system data sources. On all platforms, file data sources, which are simply text files, can be stored locally or on a network computer, and are accessible to other machines. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a connection string" for an alphabetical list of driver connection string attributes and their initial default values.

See also

Using a connection string on page 47

Configuring the product on UNIX/Linux

UNIX®

This chapter contains specific information about using your driver in the UNIX and Linux environments.

See "Environment variables" for additional platform information.

See also

Environment variables on page 36

Environment variables

The first step in setting up and configuring the driver for use is to set several environment variables. The following procedures require that you have the appropriate permissions to modify your environment and to read, write, and execute various files. You must log in as a user with full r/w/x permissions recursively on the entire Progress DataDirect *for* ODBC installation directory.

Library search path

The library search path variable can be set by executing the appropriate shell script located in the ODBC home directory. From your login shell, determine which shell you are running by executing:

```
echo $SHELL
```

C shell login (and related shell) users must execute the following command before attempting to use ODBC-enabled applications:

```
source ./odbc.csh
```

Bourne shell login (and related shell) users must initialize their environment as follows:

```
. ./odbc.sh
```

Executing these scripts sets the appropriate library search path environment variable:

- LD_LIBRARY_PATH on HP-UX IPF, Linux, and Oracle Solaris
- LIBPATH on AIX

The library search path environment variable must be set so that the ODBC core components and drivers can be located at the time of execution. After running the setup script, execute:

```
env
```

to verify that the *installation_directory*/lib directory has been added to your shared library path.

ODBCINI

Setup installs in the product installation directory a default system information file, named odbc.ini, that contains data sources. See "Data source configuration on UNIX/Linux" for an explanation of the odbc.ini file. The system administrator can choose to rename the file and/or move it to another location. In either case, the environment variable ODBCINI must be set to point to the fully qualified path name of the odbc.ini file.

For example, to point to the location of the file for an installation on /opt/odbc in the C shell, you would set this variable as follows:

```
setenv ODBCINI /opt/odbc/odbc.ini
```

In the Bourne or Korn shell, you would set it as:

```
ODBCINI=/opt/odbc/odbc.ini;export ODBCINI
```

As an alternative, you can choose to make the odbc.ini file a hidden file and not set the ODBCINI variable. In this case, you would need to rename the file to .odbc.ini (to make it a hidden file) and move it to the user's $HOME directory.

The driver searches for the location of the odbc.ini file as follows:

1. The driver checks the ODBCINI variable
2. The driver checks $HOME for .odbc.ini

If the driver does not locate the system information file, it returns an error.

See also

Data source configuration on UNIX/Linux on page 39

ODBCINST

Setup installs in the product installation directory a default file, named odbcinst.ini, for use with DSN-less connections. See "DSN-Less Connections" for an explanation of the odbcinst.ini file. The system administrator can choose to rename the file or move it to another location. In either case, the environment variable ODBCINST must be set to point to the fully qualified path name of the odbcinst.ini file.

For example, to point to the location of the file for an installation on /opt/odbc in the C shell, you would set this variable as follows:

```
setenv ODBCINST /opt/odbc/odbcinst.ini
```

In the Bourne or Korn shell, you would set it as:

```
ODBCINST=/opt/odbc/odbcinst.ini;export ODBCINST
```

As an alternative, you can choose to make the `odbcinst.ini` file a hidden file and not set the `ODBCINST` variable. In this case, you would need to rename the file to `.odbcinst.ini` (to make it a hidden file) and move it to the user's $HOME directory.

The driver searches for the location of the `odbcinst.ini` file as follows:

1. The driver checks the `ODBCINST` variable
2. The driver checks $HOME for `.odbcinst.ini`

If the driver does not locate the `odbcinst.ini` file, it returns an error.

See also

DSN-less connections on page 42

DD_INSTALLDIR

This variable provides the driver with the location of the product installation directory so that it can access support files. `DD_INSTALLDIR` must be set to point to the fully qualified path name of the installation directory.

For example, to point to the location of the directory for an installation on `/opt/odbc` in the C shell, you would set this variable as follows:

```
setenv DD_INSTALLDIR /opt/odbc
```

In the Bourne or Korn shell, you would set it as:

```
DD_INSTALLDIR=/opt/odbc;export DD_INSTALLDIR
```

The driver searches for the location of the installation directory as follows:

1. The driver checks the `DD_INSTALLDIR` variable
2. The driver checks the `odbc.ini` or the `odbcinst.ini` files for the InstallDir keyword (see "Configuration Through the System Information (`odbc.ini`) File" for a description of the InstallDir keyword)

If the driver does not locate the installation directory, it returns an error.

The next step is to test load the driver.

See also

Configuration through the system information (odbc.ini) file on page 39

The test loading tool

The second step in preparing to use a driver is to test load it.

The ivtestlib (32-bit driver) and ddtestlib (64-bit driver) test loading tools are provided to test load drivers and help diagnose configuration problems in the UNIX and Linux environments, such as environment variables not correctly set or missing database client components. This tool is installed in the `/bin` subdirectory in the product installation directory. It attempts to load a specified ODBC driver and prints out all available error information if the load fails.

For example, if the driver is installed in `/opt/odbc/lib`, the following command attempts to load the 32-bit driver on Linux, where *xx* represents the version number of the driver:

```
ivtestlib /opt/odbc/lib/ivgbqxx.so
```

Note: On Solaris, AIX, and Linux, the full path to the driver does not have to be specified for the tool. The HP-UX version, however, requires the full path.

If the load is successful, the tool returns a success message along with the version string of the driver. If the driver cannot be loaded, the tool returns an error message explaining why.

See "Version string information" for details about version strings.

The next step is to configure a data source through the system information file.

See also

Version string information on page 19

Data source configuration on UNIX/Linux

In the UNIX and Linux environments, a system information file is used to store data source information. Setup installs a default version of this file, called `odbc.ini`, in the product installation directory. This is a plain text file that contains data source definitions.

Configuration through the system information (odbc.ini) file

To configure a data source manually, you edit the `odbc.ini` file with a text editor. The content of this file is divided into three sections.

At the beginning of the file is a section named `[ODBC Data Sources]` containing *`data_source_name=installed-driver`* pairs, for example:

```
Google BigQuery=DataDirect 8.0 Google BigQuery
```

The driver uses this section to match a data source to the appropriate installed driver.

The `[ODBC Data Sources]` section also includes data source definitions. The default `odbc.ini` contains a data source definition for the driver. Each data source definition begins with a data source name in square brackets, for example, `[Google BigQuery 2]`. The data source definitions contain connection string *attribute=value* pairs with default values. You can modify these values as appropriate for your system. "Connection option descriptions" describes these attributes. See "Sample default odbc.ini file" for sample data sources.

The second section of the file is named `[ODBC File DSN]` and includes one keyword:

```
[ODBC File DSN]
DefaultDSNDir=
```

This keyword defines the path of the default location for file data sources (see "File data sources").

Note: This section is not included in the default `odbc.ini` file that is installed by the product installer. You must add this section manually.

The third section of the file is named `[ODBC]` and includes several keywords, for example:

```
[ODBC]
IANAAppCodePage=4
InstallDir=/opt/odbc
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

The IANAAppCodePage keyword defines the default value that the UNIX/Linux driver uses if individual data sources have not specified a different value. See IANAAppCodePage on page 90 in "Connection option descriptions" for details. The default value is 4.

For supported code page values, refer to "Code page values" in the *Progress DataDirect for ODBC Drivers Reference*.

The InstallDir keyword must be included in this section. The value of this keyword is the path to the installation directory under which the `/lib` and `/locale` directories are contained. The installation process automatically writes your installation directory to the default `odbc.ini` file.

For example, if you choose an installation location of `/opt/odbc`, then the following line is written to the `[ODBC]` section of the default `odbc.ini`:

```
InstallDir=/opt/odbc
```

Note: If you are using only DSN-less connections through an `odbcinst.ini` file and do not have an `odbc.ini` file, then you must provide `[ODBC]` section information in the `[ODBC]` section of the `odbcinst.ini` file. The driver and Driver Manager always check first in the `[ODBC]` section of an `odbc.ini` file. If no `odbc.ini` file exists or if the `odbc.ini` file does not contain an `[ODBC]` section, they check for an `[ODBC]` section in the `odbcinst.ini` file. See "DSN-less connections" for details.

ODBC tracing allows you to trace calls to the ODBC driver and create a log of the traces for troubleshooting purposes. The following keywords all control tracing: Trace, TraceFile, TraceDLL, ODBCTraceMaxFileSize, and ODBCTraceMaxNumFiles.

For a complete discussion of tracing, refer to "ODBC trace" in the *Progress DataDirect for ODBC Drivers Reference*.

See also

Connection option descriptions on page 69
Sample default odbc.ini file on page 40
File data sources on page 43
IANAAppCodePage on page 90
DSN-less connections on page 42

Sample default odbc.ini file

The following is a sample `odbc.ini` file that the installer program installs in the installation directory. All occurrences of ODBCHOME are replaced with your installation directory path during installation of the file. Values that you must supply are enclosed by angle brackets (< >). If you are using the installed `odbc.ini` file, you must supply the values and remove the angle brackets before that data source section will operate properly. Commented lines are denoted by the # symbol. This sample shows a 32-bit driver with the driver file name beginning with `iv`. A 64-bit driver file would be identical except that driver name would begin with `dd` and the list of data sources would include only the 64-bit drivers.

```
[ODBC Data Sources]
Google BigQuery=DataDirect 8.0 Google BigQuery

[Google BigQuery]
Driver=ODBCHOME/lib/ivgbq28.so
Description=DataDirect 8.0 Google BigQuery
AccessToken=abcdefghi12345678
AllowLargeResults=false
ApplicationUsingThreads=1
AuthenticationMethod=oauth2.0
AuthURI=https://accounts.google.com/o/oauth2/auth
```

```
ClientID=123abc.apps.googleusercontent.com
ClientSecret=ab123xy
ConfigOptions=
ConnectionReset=0
CreateMap=2
Dataset=mydataset
DataSourceName=
Description=
EnableLoginPrompt=0
FetchSize=100
IANAAppCodePage=
JobTimeout=0
JsonFormat=raw
JVMArgs=-Xmx256m
JVMClasspath=
JWTAudience=https://accounts.google.com/o/oauth2/token
KMSKeyName=
LegacyDataset=_queries_
LegacyTable=_sql_*
Location=
LoadBalanceTimeout=0
LogConfigFile=
LoginTimeout=15
MaximumBytesBilled=0
MaxPoolSize=100
MinPoolSize=0
PrimaryKeyPattern=
Project=myproject
Pooling=0
ProxyHost=
ProxyPassword=
ProxyPort=
ProxyUser=
RedirectURI=
RefreshSchema=0
RefreshSchemaForDDL=true
RefreshToken=wxyz123456789
ReportCodepageConversionErrors=0
RetryExceptions=0
SchemaMap=
Scope=https://www.googleapis.com/auth/bigquery
ServiceAccountEmail=
ServiceAccountPrivateKey=
SQLEngineMode=2
Syntax=
TokenURI=https://accounts.google.com/o/oauth2/token
UseQueryCache=1
UseStreamingInsert=1
WSFetchSize=1000000
WSPoolSize=1
WSRetryCount=5
WSTimeout=120

[ODBC]
IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
[ODBC File DSN]
DefaultDSNDir=
UseCursorLib=0
```

To modify or create data sources in the `odbc.ini` file, use the following procedures.

- **To modify a data source:**

a) Using a text editor, open the `odbc.ini` file.

b) Modify the default attributes in the data source definitions as necessary based on your system specifics.

Consult "Connection option descriptions" for the Google BigQuery attributes.

c) After making all modifications, save the `odbc.ini` file and close the text editor.

Important: The "Connection option descriptions" section lists both the long and short names of the attribute. When entering attribute names into `odbc.ini`, you must use the long name of the attribute. The short name is not valid in the `odbc.ini` file.

- **To create a new data source:**

a) Using a text editor, open the `odbc.ini` file.

b) Copy an appropriate existing default data source definition and paste it to another location in the file.

c) Change the data source name in the copied data source definition to a new name. The data source name is between square brackets at the beginning of the definition, for example, `[GoogleBigQuery]`.

d) Modify the attributes in the new definition as necessary based on your system specifics.

Consult "Connection option descriptions" for the Google BigQuery attributes.

e) In the `[ODBC]` section at the beginning of the file, add a new *data_source_name=installed-driver* pair containing the new data source name and the appropriate installed driver name.

f) After making all modifications, save the `odbc.ini` file and close the text editor.

Important: The "Connection option descriptions" lists both the long and short name of the attribute. When entering attribute names into `odbc.ini`, you must use the long name of the attribute. The short name is not valid in the `odbc.ini` file.

See also

Connection option descriptions on page 69

DSN-less connections

Connections to a data source can be made via a connection string without referring to a data source name (DSN-less connections). This is done by specifying the "`DRIVER=`" keyword instead of the "`DSN=`" keyword in a connection string, as outlined in the ODBC specification. A file named `odbcinst.ini` must exist when the driver encounters `DRIVER=` in a connection string.

Setup installs a default version of this file in the product installation directory (see "ODBCINST" for details about relocating and renaming this file). This is a plain text file that contains default DSN-less connection information. You should not normally need to edit this file. The content of this file is divided into several sections.

At the beginning of the file is a section named `[ODBC Drivers]` that lists installed drivers, for example,

```
DataDirect 8.0 Google BigQuery=Installed
```

This section also includes additional information for each driver.

The final section of the file is named [ODBC]. The [ODBC] section in the odbcinst.ini file fulfills the same purpose in DSN-less connections as the [ODBC] section in the odbc.ini file does for data source connections. See "Configuration through the system information (odbc.ini) file" for a description of the other keywords this section.

Note: The odbcinst.ini file and the odbc.ini file include an [ODBC] section. If the information in these two sections is not the same, the values in the odbc.ini [ODBC] section override those of the odbcinst.ini [ODBC] section.

See also

ODBCINST on page 37
Configuration through the system information (odbc.ini) file on page 39

Sample odbcinst.ini File

The following is a sample odbcinst.ini. All occurrences of ODBCHOME are replaced with your installation directory path during installation of the file. Commented lines are denoted by the # symbol. This sample shows a 32-bit driver with the driver file name beginning with iv; a 64-bit driver file would be identical except that driver names would begin with dd.

```
[ODBC Drivers]
DataDirect 8.0 Google BigQuery=Installed

[DataDirect 8.0 Google BigQuery]
Driver=ODBCHOME/lib/ivgbc28.so
JarFile=ODBCHOME/java/lib/googlebigquery.jar
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/Help/GoogleBigQueryHelp
SQLLevel=0
UsageCount=1

[ODBC]
#This section must contain values for DSN-less connections
#if no odbc.ini file exists. If an odbc.ini file exists,
#the values from that [ODBC] section are used.

IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

File data sources

The Driver Manager on UNIX and Linux supports file data sources. The advantage of a file data source is that it can be stored on a server and accessed by other machines, either Windows, UNIX, or Linux. See "Configuring and connecting to data sources" for a general description of ODBC data sources on both Windows and UNIX.

A file data source is simply a text file that contains connection information. It can be created with a text editor. The file normally has an extension of .dsn.

For example, a file data source for the driver would be similar to the following:

```
[ODBC]
Driver=DataDirect 8.0 Google BigQuery
Project=myproject
Dataset=mydataset
AccessToken=abcdefghi12345678
RefreshToken=wxyz123456789
ClientID=123abc.apps.googleusercontent.com
ClientSecret=ab123xy
```

It must contain all basic connection information plus any optional attributes. Because it uses the DRIVER= keyword, an odbcinst.ini file containing the driver location must exist (see "DSN-less connections").

The file data source is accessed by specifying the FILEDSN= instead of the DSN= keyword in a connection string, as outlined in the ODBC specification. The complete path to the file data source can be specified in the syntax that is normal for the machine on which the file is located. For example, on Windows:

```
FILEDSN=C:\Program Files\Common Files\ODBC\DataSources\GoogleBigQuery2.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/GoogleBigQuery2.dsn
```

If no path is specified for the file data source, the Driver Manager uses the DefaultDSNDir property, which is defined in the [ODBC File DSN] setting in the odbc.ini file to locate file data sources (see "Data source configuration on UNIX/Linux" for details). If the [ODBC File DSN] setting is not defined, the Driver Manager uses the InstallDir setting in the [ODBC] section of the odbc.ini file. The Driver Manager does not support the SQLReadFileDSN and SQLWriteFileDSN functions.

As with any connection string, you can specify attributes to override the default values in the data source:

```
FILEDSN=/home/users/john/filedsn/GoogleBigQuery2.dsn;Project=myproject;Dataset=mydataset
```

See also

Configuring and connecting to data sources on page 36
DSN-less connections on page 42
Data source configuration on UNIX/Linux on page 39

UTF-16 applications on UNIX and Linux

Because the DataDirect Driver Manager allows applications to use either UTF-8 or UTF-16 Unicode encoding, applications written in UTF-16 for Windows platforms can also be used on UNIX and Linux platforms.

The Driver Manager assumes a default of UTF-8 applications; therefore, two things must occur for it to determine that the application is UTF-16:

- The definition of SQLWCHAR in the ODBC header files must be switched from "char *" to "short *". To do this, the application uses #define SQLWCHARSHORT.
- The application must set the encoding for the environment or connection using one of the following attributes. If your application passes UTF-8 encoded strings to some connections and UTF-16 encoded strings to other connections in the same environment, encoding should be set for the connection only; otherwise, either method can be used.
  - To configure the encoding for the environment, set the ODBC environment attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16, for example:

    ```
    rc = SQLSetEnvAttr(*henv,
    SQL_ATTR_APP_UNICODE_TYPE,(SQLPOINTER)SQL_DD_CP_UTF16, SQL_IS_INTEGER);
    ```

- To configure the encoding for the connection only, set the ODBC connection attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16. For example:

  ```
  rc = SQLSetConnectAttr(hdbc, SQL_ATTR_APP_UNICODE_TYPE, SQL_DD_CP_UTF16,
  SQL_IS_INTEGER);
  ```

Setting the library path environment variable (Windows)

Before you can use your driver, you must set the PATH environment variable to include the path of the `jvm.dll` file of your Java™ Virtual Machine (JVM).

Note: The installer program sets the PATH environment variable to include the path of the `jvm.dll` file by default.

Configuring data sources with the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

The driver includes an enhanced setup dialog, the Progress DataDirect Google BigQuery Configuration Manager, that allows you to configure data sources, generate connection strings, test connections, and execute test queries. On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using the Configuration Manager, as described in this section.

Note: As you configure your data source, the Configuration Manager generates a corresponding connection string in the **Connection String** pane. To use your connection string, click the Copy button () and paste the string to a location that can be used by your application. See "Generating connection strings with the Configuration Manager" for details.

Note: Connection string attributes can be used to override the default values of the data source if you want to change these values at connection time.

To configure and test a data source:

1. Open the Windows ODBC Administrator.
2. Open the Configuration Manager through the **User DSN** or **System DSN** tab.

   - **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.

   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the Configuration Manager.

Note: Configuring a new data source using the File DSN tab is not currently supported with the configuration manager.

The Google BigQuery Configuration Manager window opens.

3. From the Configuration Manager window, provide values of the connection options you want to configure in the corresponding fields. To view more options, select the tabs at the top of the page. See "Connection option descriptions" for descriptions of the supported options.

   Note: See "Connection string examples" for a list of required options used for different configurations. The options and settings described in that section apply to all methods of configuration.

4. At any point during the process, you can click **Test Connect** to attempt to connect to the instance with your settings. In the Test Connection window:
   a) Provide values for any fields required by your instance. Note that the information you enter in the logon dialog box during a test connect is not saved.
   b) Optionally, in the **Test Query** field, enter any SQL queries you want to execute during the test. For example:

      ```
      SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
      ```

   c) Click **Execute**.

   Important: An initial connection may take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service. Similar delays may occur depending on the Create Map (CreateMap) setting. For example, a delay may be incurred if Create Map is set to 4 (OnChange), the default, and changes have been made to the schema on the Microsoft SharePoint backend. For details, see "Create Map".

   If successful, the window displays a confirmation message and, if a query was specified, the results of the query.
5. Click **Save** to apply your values as the default when connecting with the data source.

Generating connection strings with the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

The Progress DataDirect Google BigQuery Configuration Manager supports generating connection strings that can be used with your application. To generate a connection string, create a data source as described in "Configuring data sources with the Configuration Manager." As you provide connection option values, the Configuration Manager generates a connection string in the **Connection String** pane that corresponds to the data source.

In addition to providing values for connection option fields, you can manually edit your string by clicking the Edit button (). Note that editing your connection string also changes the values for the data source.

After you are done configuring the connection options, click **Test Connect** to test your connection string. See "Testing connections and queries with the Configuration Manager" for more information.

To use your string, click the Copy button () and paste the string to a location that can be used by your application.

Using a connection string

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because the information is not stored in a data source.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=value]...]
```

"Connection option descriptions" lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Google BigQuery for Linux/UNIX/Windows is:

```
DSN=GoogleBigQuery;PJT=myproject;DS=mydataset;AT=abcdefghi12345678;
RT=wxyz123456789;CI=123abc.apps.googleusercontent.com;CS=ab123xy
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=GoogleBigQuery.dsn;PJT=myproject;DS=mydataset;AT=abcdefghi12345678;
RT=wxyz123456789;CI=123abc.apps.googleusercontent.com;CS=ab123xy
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 8.0 Google BigQuery;PJT=myproject;DS=mydataset;
AT=abcdefghi12345678;RT=wxyz123456789;CI=123abc.apps.googleusercontent.com;
CS=ab123xy
```

See also

Connection option descriptions on page 69

Testing connections and queries with the Configuration Manager

Note: The Configuration Manager is currently supported only on Windows platforms.

You can quickly test data sources, connection strings and queries using Progress DataDirect Google BigQuery Configuration Manager.

To test your connection and query:

1. Open the Windows ODBC Administrator. Then, select or add a data source to open the Google BigQuery Configuration Manager.

   - Windows: Start the ODBC Administrator by selecting its icon from the Progress DataDirect program group. To use an existing data source, select one to configure from the ODBC Administrator; otherwise, add a new data source to test new configurations or connection strings.

   For detailed information on launching the Configuration Manager, see "Configuring data sources with the Configuration Manager."

2. If you are not testing an existing data source, provide connection information using one of the following methods:

   - Enter a connection string you provide by clicking the Edit button (); then, pasting your string into the Connection String field. If you prefer, you can also type a string directly into this field.
   - Enter values of the connection options you want to configure into the corresponding fields. The Google BigQuery Configuration Manager will generate a data source and connection string based on the values you specify.

3. Click **Test Connect** to attempt to connect to the instance using the string specified in the Connection String field. The **Test Connection** window appears.
4. Provide connection option values for any fields required by your instance.
5. Optionally, to execute a test query with the test connection, enter a SQL query into the Test Query field. For example:

   ```
   SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
   ```

6. Click **Execute**.

   Important: An initial connection may take a few minutes, depending on network speeds and the amount of metadata the driver must retrieve from the service.

   If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

Password Encryption Tool (UNIX/Linux only)

On UNIX and Linux, Progress DataDirect provides a Password Encryption Tool, called `ddencpwd`, that encrypts passwords for secure handling in connection strings and `odbc.ini` files. At connection, the driver decrypts these passwords and passes them to the data source as required. Passwords can be encrypted for any option, including:

- KeyPassword
- KeyStorePassword
- TrustStorePassword
- Password

To use the Password Encryption Tool:

1. From a command line, navigate to the directory containing the ddencpwd application. By default, this is *install_directory*/tools.
2. Enter the following command:

   ddencpwd *password*

   where:

   password

   is the password you want to encrypt.

3. The tool returns an encrypted password value. Specify the returned value for the corresponding attribute in the connection string or odbc.ini file. For example, if you encrypted the password for KeyPassword, specify the following in your connection string or datasource definition:

   KeyPassword=*returned_value*

4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an odbc.ini file, save your file.

This completes this tutorial. You are now ready to connect using encrypted passwords.

Using a logon dialog box

Some ODBC applications display a logon dialog box when you are connecting to a data source. To connect, provide the values described in the following sections; then, click **OK** to complete the logon.

In the dialog box, provide the following information depending on the authentication method:

- In the Client ID field, type the client ID key for your application.
- In the Client Secret field, type the client secret for your application.
- In the Access Token, type theaccess token used to authenticate to your service.
- In the Refresh Token field, type the refresh token used to connect to your service.
- In the Scope field, type a space-separated list of OAuth scopes.

Connecting through a proxy server

In some environments, your application may need to connect through a proxy server, for example, if your application accesses an external resource such as a Web service. At a minimum, your application needs to provide the following connection information when you invoke the JVM if the application connects through a proxy server:

- Server name or IP address of the proxy server
- Port number on which the proxy server is listening for HTTP/HTTPS requests

In addition, if authentication is required, your application may need to provide a valid user ID and password for the proxy server. Consult with your system administrator for the required information.

For example, the following command invokes the JVM while specifying a proxy server named `pserver`, a port of `808`, and provides a user ID and password for authentication:

```
java -Dhttp.proxyHost=pserver -Dhttp.proxyPort=808 -Dhttp.proxyUser=smith
-Dhttp.proxyPassword=secret -cp sforce.jar com.acme.myapp.Main
```

Alternatively, you can use the Proxy Host, Proxy Port, Proxy User, and Proxy Password connection attributes. See "Connection option descriptions" for details about these attributes.

See also

Connection option descriptions on page 69

Using security

The driver supports the following security features:

- *Authentication* is the process of identifying a user.
- *Data encryption* is the conversion of data into a form that cannot be easily understood by unauthorized users. This is enabled by default.

Authentication

Authentication ensures that only the authorized users are allowed to connect to a Google BigQuery instance.

The Google BigQuery driver supports the following methods of authentication:

- *OAuth 2.0 authentication* allows the user to authenticate to a Google BigQuery instance without having to specify user ID and password.
- *Service account authentication*. It allows the users to authenticate to a Google BigQuery instance using a service account, which is a type of Google account that represents an application instead of an individual end user.

Configuring OAuth 2.0 authentication

The driver supports OAuth 2.0, which is an open protocol for token-based authentication. OAuth 2.0 allows you to authenticate without specifying a user ID or password, eliminating the risk of exposing them to unauthorized access. It uses an access token, accompanied by the values discussed below, to authenticate to a Google BigQuery instance. See "Generating access token and refresh token" to know how to obtain an access token.

To configure the driver to use OAuth 2.0 authentication, set the following connection options:

- Set the Authentication Method option to `oauth2.0`.
- Set at least one of the following options:
  - Access Token: Set this to specify the access token you have obtained to authenticate to Google BigQuery.
  - Refresh Token: Set this to specify the refresh token you have obtained to authenticate to Google BigQuery.

If a value for the Access Token option is not specified, the driver uses the value of the Refresh Token option to make a connection. If both values are not specified, the driver cannot make a successful connection. If both are specified, the driver uses the Access Token value; however, if the Access Token value expires, it uses the Refresh Token value to generate a new Access Token value.

- Set the Client ID option to specify the consumer key for your application.
- Set the Client Secret option to specify the consumer secret for your application.
- Optionally, set the Auth URI option to specify the endpoint for obtaining an authorization code from a private endpoint. The default value is `https://accounts.google.com/o/oauth2/auth`.
- Optionally, set the Token URI option to specify the endpoint for retrieving access tokens. The default value is `https://accounts.google.com/o/oauth2/token`.
- Optionally, set the Scope option to specify the OAuth scope. It limits the permissions granted by an access token.

The following examples show how to connect to a Google BigQuery instance using OAuth2.0 authentication.

Using a connection string:

```
DRIVER=DataDirect 8.0 Google BigQuery;AuthenticationMethod=oauth2.0;
Project=myproject;Dataset=mydataset;
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy
```

Using the `odbc.ini` file:

```
Driver=ODBCHOME/lib/xxgbq28.yy
AuthenticationMethod=oauth2.0
Project=myproject
Dataset=mydataset
AccessToken=abcdefghi12345678
RefreshToken=wxyz123456789
ClientID=123abc.apps.googleusercontent.com
ClientSecret=ab123xy
```

See also

Generating access tokens and refresh tokens on page 55
Authentication Method on page 78
Access Token on page 76
Refresh Token on page 107
Client ID on page 80
Schema Map on page 109
Client Secret on page 80

Dynamic authorization code grant

A dynamic authorization code grant allows you to initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or third-party application. Similar to authorization code grant, dynamic authorization code grant is typically used for web and native applications. It also provides secure connections by requiring multiple points of authentication before permitting access to data.

When connecting with dynamic authorization code grant flow, the driver launches the login prompt for your service in a separate browser window. After you submit your user and password credentials via the prompt, the driver exchanges your login credentials and client credentials for the Authorization Code from the location specified by the Authorization URI option. The driver then navigates to the endpoint specified by the Token URI option to exchange the authorization code for the access and refresh tokens. Finally, the application is redirected to the location provided in the Redirect URI option to begin the session.

After the grant flow is complete, the driver continues to use the access and refresh tokens to access data resources for the lifetime of the ODBC connection or until both the access and refresh tokens expire, whichever occurs first. If both tokens expire while the connection is still active, the driver launches the login prompt to reinitiate the flow.

Note: The dynamic authorization grant requires the manual submission of login credentials via the login prompt for your service; therefore, the driver does not support dynamic authorization grant in headless environments.

To use dynamic authorization code grant:

- Set the Authentication Method (`AuthenticationMethod`) option to `oauth2.0`.
- Set the Enable Login Prompt (`EnableLoginPrompt`) option to `1` (enabled). When Enable Login Prompt is enabled, the driver launches the login prompt for your service in a separate browser window to initiate the OAuth grant flow.
- Set the SQL Engine Mode (`SQLEngineMode`) option to `2` (Direct).

  Note: The dynamic authorization code grant is supported only in Direct mode.

- Set the Client ID (`ClientID`) option to specify the client ID key for your application.
- Set the Client Secret (`ClientSecret`) option to specify the client secret for your application.

  Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

- Set the Authorization URI (`AuthURI`) option to specify the endpoint for obtaining an authorization code.
- Set the Token URI (`TokenURI`) option to specify the endpoint used to exchange authentication credentials for access tokens.
- Set the Scope (`Scope`) option to specify a space-separated list of OAuth scopes to limit the permissions granted by the access token.
- Set the Redirect URI (`RedirectURI`) option to specify the endpoint that the client is returned to after authenticating with a third-party service. Note that the value of the Redirect URI (RedirectURI) option must include the port number. For example, `RedirectURI=http://localhost:80` or `RedirectURI=http://localhost:8080`.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

The following example demonstrates a basic session using the dynamic authorization code grant:

Connection URL:

```
Driver=DataDirect 8.0 Google BigQuery;
AuthenticationMethod=oauth2.0;EnableLoginPrompt=1;SQLEngineMode=2;
ClientID=abcdefghij1k2lmn3o4p5qr67s;ClientSecret=FaZBFRsGXTaR;
AuthURI=https://accounts.google.com/o/oauth2/auth;
TokenURI=https://accounts.google.com/o/oauth2/token;
Scope=https://www.googleapis.com/auth/bigquery;
RedirectURI=http://localhost:80;
```

`odbc.ini` File:

```
Driver=ODBCHOME/lib/ivgbq28.so
...
Description=DataDirect 8.0 Google BigQuery
...
AuthenticationMethod=oauth2.0
...
EnableLoginPrompt=1
...
SQLEngineMode=2
...
ClientID=abcdefghij1k2lmn3o4p5qr67s
...
ClientSecret=FaZBFRsGXTaR
...
AuthURI=https://accounts.google.com/o/oauth2/auth
...
TokenURI=https://accounts.google.com/o/oauth2/token
...
Scope=https://www.googleapis.com/auth/bigquery
...
RedirectURI=http://localhost:80
...
```

See also

Connection option descriptions on page 69

Configuring service account authentication

The driver supports service account authentication. A service account is a type of Google account that represents an application instead of an individual end user. Unlike a user account, a service account allows your application to authenticate and communicate to Google APIs without direct human intervention. This is useful for applications that need to access their own data, not the user's data. For a successful service account authentication, you need:

- **Private key file** or **Private key**
  - The private key file is a `.json` or `.p12` file that contains the key required to authenticate API calls. You can download it from the Google Cloud Platform (GCP) Console.
  - The private key is contained in the private key file downloaded from the GCP Console.
- **Service account's email address**: A unique email address that is provisioned while creating a service account.

To know more about service account authentication, refer to the Google documentation.

To configure the driver to use service account authentication, set the following connection options:

- Set the Authentication Method option to `serviceaccount`.
- Set the Service Account Email option to specify your service account's email address.

- Set either the Service Account Key Content option or the Service Account Private Key option
  - The Service Account Key Content option specifies the private key required to authenticate to Google BigQuery. Use this option if you do not want to persist the private key file in your environment.

    The value of the Service Account Key Content option should be protected for security reasons. The option has been intentionally excluded from the Windows Setup dialog. In addition, the value should not be hardcoded in an ODBC data source. Rather, it may be specified in the application with the SQLConnect ODBC API, in a connection string as with SQLDriverConnect and SQLBrowseConnect, or through the Logon dialog prompt. (If your operating system supports the Logon dialog, the prompt appears when the value has not been otherwise specified.)
  - The Service Account Private Key option specifies the full path to the `.json` or `.p12` file that contains the private key. The driver extracts the private key value from the specified file and uses it to authenticate the user to the database. Use this option if it is preferable to persist the private key file.
- Optionally, set the JWT Audience option to specify the JWT audience claim associated with your service account. The default value is `https://accounts.google.com/o/oauth2/token`.
- Optionally, set the Token URI option to specify the endpoint for retrieving access tokens. The default value is `https://accounts.google.com/o/oauth2/token`.

The following examples show how to connect to a Google BigQuery instance using service account authentication.

Service Account Key Content option in a connection string:

```
DRIVER=DataDirect 8.0 Google BigQuery;AuthenticationMethod=serviceaccount;
Project=myproject;Dataset=mydataset;ServiceAccountEmail=abc123@iam.gserviceaccount.com;
ServiceAccountKeyContent=NJJXZexIHJFGYBgkqhkiG9w0BAQnWRwiHANpf3MC1pVRqhtTE5tSpxZeQnICG
4zp087Eidn4qc66udg8KAHknyqFdj7b\n+MgxMFPAvJ59cylHFaHA4pGmeGfVqzYub6LEs9aN/751jmZqcuAYp
5nXRF1EvJPN\nsDuJGLvuuDBZW0iux0liEHmcQVBBKwIx8t+EQxePGTiLsBoCdzOUsi4UWWv\nASqfdP/kSX+N;);
```

Service Account Key Content and the `odbc.ini` file:

Note: For security reasons, the value of the Service Account Key Content option should not be specified in the `odbc.ini` file.

```
Driver=ODBCHOME/lib/xxgbq28.yy
AuthenticationMethod=serviceaccount
Project=myproject
Dataset=mydataset
ServiceAccountEmail=abc123@iam.gserviceaccount.com
```

Service Account Private Key option in a connection string:

```
DRIVER=DataDirect 8.0 Google BigQuery;AuthenticationMethod=serviceaccount;
Project=myproject;Dataset=mydataset;ServiceAccountEmail=abc123@iam.gserviceaccount.com;
ServiceAccountPrivateKey=abc123.json;);
```

Service Account Private Key in the `odbc.ini` file:

```
Driver=ODBCHOME/lib/xxgbq28.yy
AuthenticationMethod=serviceaccount
Project=myproject
Dataset=mydataset
ServiceAccountEmail=abc123@iam.gserviceaccount.com
ServiceAccountPrivateKey=abc123.json
```

See also

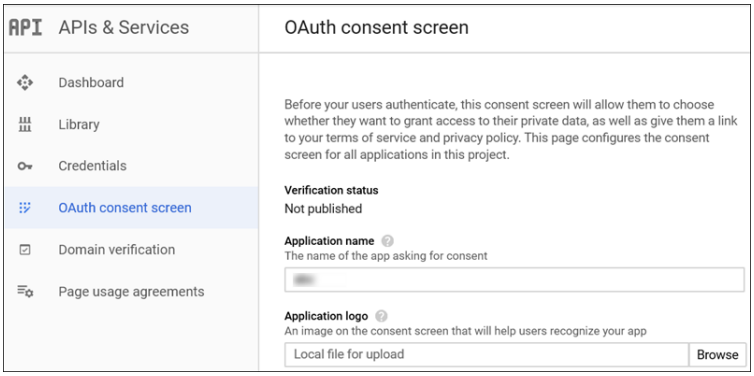
Authentication Method on page 78
Service Account Email on page 112
Service Account Key Content on page 112
Service Account Private Key on page 113
Connection option descriptions on page 69

Generating access tokens and refresh tokens

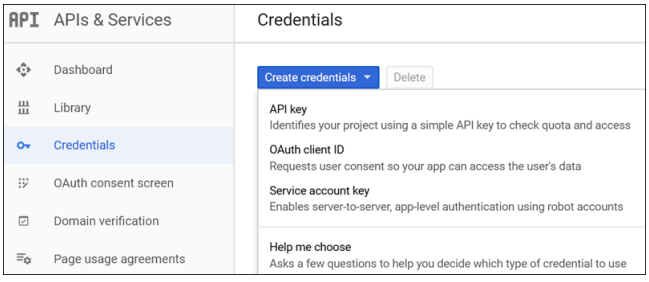
This section guides you through the process of generating access tokens and refresh tokens that are used to authenticate to Google BigQuery when OAuth 2.0 is set as the authentication method (`AuthenticationMethod=oauth2.0`).

To generate an access token and a refresh token from Google Console:

1. Click the following link to navigate to Google Developer Console: https://console.developers.google.com/.
2. Enter your login credentials; then, click **Next**.
3. On the left pane, click **OAuth consent screen**.



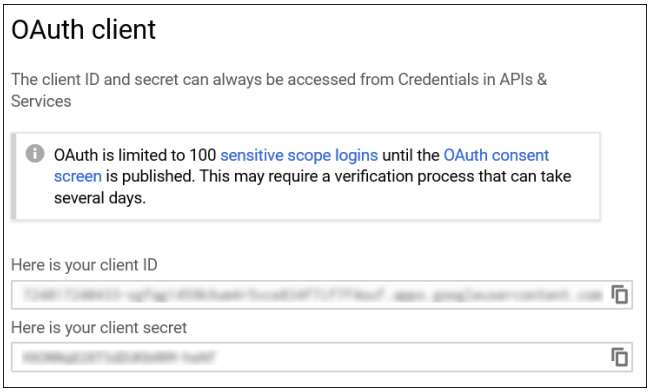
4. On the OAuth consent screen, enter the following details; then, click **Save**:
   - Application name
   - Support email
   - Scopes for Google APIs
   - Authorized domains
   - Application Homepage link
   - Application Privacy Policy link
5. On the left pane, click **Credentials**. Next, from the Create credentials drop-down list, select **OAuth client ID**.



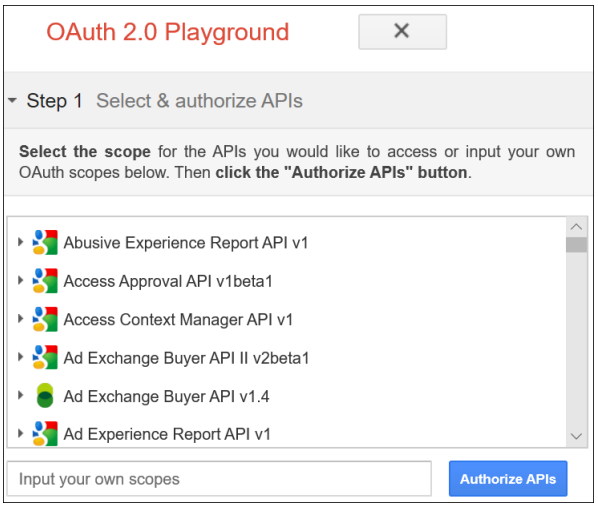
6. The Create OAuth client ID screen appears. Select **Other** as your Application type, enter your application's name, and then click **Create**.

   .

7. OAuth client window appears. Copy and save your client ID and client secret from the corresponding fields. You will need them later in this procedure. Note that they can also be accessed from the Credentials screen, if necessary.



8. Click the following link to navigate to OAuth 2.0 Playground: https://developers.google.com/oauthplayground/.
9. Select the required scopes; then, click **Authorize APIs**.



   Login screen appears.

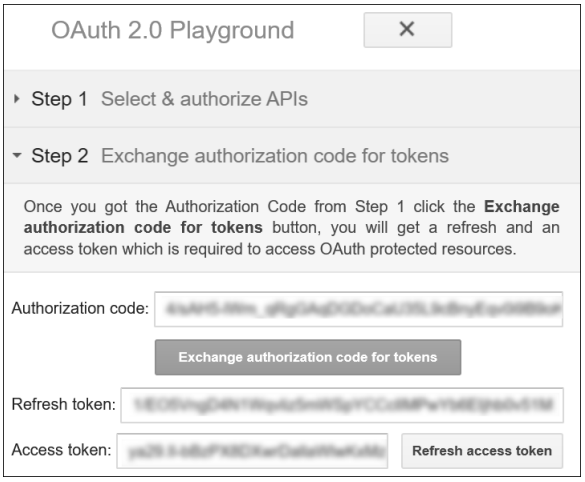
10. Click your username. The following message appears: "Google OAuth 2.0 Playground wants to access your Google Account."

11. Scroll down and click **Allow**.

    You are redirected to the OAuth 2.0 Playground. It contains the newly generated authorization code for your application.

12. Click **Exchange authorization code for tokens**.

    The refresh token and access token are generated in the corresponding fields.



See also

Configuring OAuth 2.0 authentication on page 50
Access Token on page 76
Refresh Token on page 107

Performance considerations

The connection options described in this section directly affect the performance of your driver. To tune for performance, configure your driver according to the recommended settings and your environment.

Fetch Size/Web Service Fetch Size (FetchSize/WSFetchSize): The connection options Fetch Size and Web Service Fetch Size can be used to adjust the trade-off between throughput and response time. In general, setting larger values for Web Service Fetch Size and Fetch Size will improve throughput, but can reduce response time.

For example, if an application attempts to fetch 100,000 rows from the remote data source and Web Service Fetch Size is set to 500, the driver must make 200 Web service calls to get the 100,000 rows. If, however, Web Service Fetch Size is set to 4000, the driver only needs to make 25 Web service calls to retrieve 100,000 rows. Web service calls are expensive, so generally, minimizing Web service calls increases throughput. In addition, many Cloud data sources impose limits on the number of Web service calls that can be made in a given period of time. Minimizing the number of Web service calls used to fetch data also can help prevent exceeding the data source call limits.

For many applications, throughput is the primary performance measure, but for interactive applications, such as Web applications, response time (how fast the first set of data is returned) is more important than throughput. For example, suppose that you have a Web application that displays data 50 rows to a page and that, on average, you view three or four pages. Response time can be improved by setting Fetch Size to 50 (the number of rows displayed on a page) and WSFetch Size to 200. With these settings, the driver fetches all of the rows from the remote data source that you would typically view in a single Web service call and only processes the rows needed to display the first page.

Use Storage API (UseStorageAPI): The Use Storage API connection option can be used to improve performance by enabling the driver to use the Google BigQuery Storage API when fetching large result sets. When Use Storage API is set to **enabled**, the driver uses the Storage API for selects when the number of rows in the result set exceeds the value of the Storage API Threshold option, and the number of pages in the result set exceeds the value of the Storage API Min Page Count option.

Use Streaming Insert (UseStreamingInsert): The Use Streaming Insert connection option can be used to improve performance by enabling the driver to use the Google BigQuery Streaming API for executing batch inserts. When Use Streaming Insert is set to **enabled**, the driver uses the Streaming API for insert operations with the `tabledata.insertAll` method.

Web Service Pool Size (WSPoolSize): Web Service Pool Size determines the maximum number of sessions the driver uses when there are multiple active connections to a REST service. By increasing this number, you increase the number of sessions the driver uses to distribute calls to a REST service, thereby improving throughput and performance. For example, if this option is set to 1, and you have two open connections, the session must complete a call from one connection before it can begin processing a call from the other connection. However, if Web Service Pool Size is set to 2, a second session is opened that allows calls from both connections to be processed simultaneously.

Note: The number specified for Web Service Pool Size should not exceed the amount of sessions permitted by your REST service.

Internal and external tables support

The driver supports create, read, update, and delete operations in internal tables, which are stored inside Google BigQuery, and read operations in external tables, which are stored in data sources outside Google BigQuery.

Note that Google BigQuery has some rules on how Data Definition Language (DDL) and Data Manipulation Language () statements can be used. Refer to the Google BigQuery documentation to learn more about these rules.

Google BigQuery Storage API

The driver supports the Google BigQuery Storage API for fetching large result sets. Compared to the standard BigQuery API, the BigQuery Storage API provides increased throughput and allows the driver to more efficiently manage large result sets.

Note: Currently, the Storage API is supported only on Windows 64-bit, Linux 64-bit, and JVM 64-bit. If an application attempts to use the Storage API on an unsupported platform, the driver falls back to the Standard API.

You can enable the driver to use the Storage API with the Use Storage API connection option. You can further configure how the driver uses the Storage API with the Storage API Threshold and Storage API Min Page Count connection options.

Before enabling the driver to use the Storage API, prerequisites described in the Google Cloud *BigQuery Storage API reference* must be met: https://cloud.google.com/bigquery/docs/reference/storage/.

See also

Storage API attributes on page 71
Use Storage API on page 118
Storage API Min Page Count on page 115
Storage API Threshold on page 116

Google BigQuery Streaming API

The driver supports the Google BigQuery Streaming API for executing batch inserts. The BigQuery Streaming API provides increased throughput and allows the driver to manage insert operations efficiently.

You can enable the driver to use the Streaming API with the Use Streaming Insert (`UseStreamingInsert`) connection option. When this connection option is set to `1`, the driver uses the `tabledata.insertAll` method for streaming batched inserts.

Note: When using the Streaming API for insert operations, the rows are queued and appear in the table after a delay. The duration of the delay varies and can go up to 30 minutes. You must not modify the table or run other operations on the table while there are rows in the streaming queue. Changing the table's metadata or running DML operations can cause uncommitted streaming rows to be lost.

Before enabling the driver to use the Streaming API, prerequisites described in the Google Cloud *BigQuery Streaming API reference* must be met: https://cloud.google.com/bigquery/docs/streaming-data-into-bigquery.

See also

Use Streaming Insert on page 119

DataDirect connection pooling

Connection pooling allows you to *reuse* connections rather than creating a new one every time the driver needs to establish a connection to the underlying database. Your Progress DataDirect for ODBC driver enables connection pooling without requiring changes to your client application.

Refer to "DataDirect Connection Pooling" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

To configure the driver to use connection pooling:

- Set the Connection Pooling (Pooling) option to `1`.
- Set the Connection Reset (ConnectionReset) option to `1` or `0`. Setting it to `1` resets the state of connections removed from the connection pool for reuse by an application to the initial configuration of the connection. Setting it to `0` does not reset the state of connections.
- Set the Load Balance Timeout (LoadBalanceTimeout) option to specify an integer value to specify the amount of time, in seconds, to keep connections open in a connection pool.
- Set the Max Pool Size (MaxPoolSize) option to specify an integer value to specify the maximum number of connections within a single pool.

- Set the Min Pool Size (MinPoolSize) option to an integer value to specify the minimum number of connections that are opened and placed in a connection pool when it is created.

The following examples show how to configure the driver to use connection pooling:

Connection string

```
DSN=GoogleBigQuery;Pooling=1;ConnectionReset=0;LoadBalanceTimeout=0;
MaxPoolSize=100;MinPoolSize=0;
```

odbc.ini

```
[GoogleBigQuery]
Driver=ODBCHOME/lib/xxgbq28.yy
...
Pooling=1
...
ConnectionReset=0
...
LoadBalanceTimeout=0
...
MaxPoolSize=100
...
MinPoolSize=0
...
```

Timeouts

The following types of timeout situations can occur when connecting to Google BigQuery:

- **Session timeouts**. Most remote data sources impose a limit on the duration of active sessions, meaning a session can fail with a session timeout error if the session extends past the limit. This is particularly true when connection pooling is used. The driver automatically attempts to re-establish a new session if the driver receives a session timeout error from a data source. The driver uses the initial servername, port (if appropriate), remote user ID, and remote password (encrypted) to re-establish the session. If the attempt fails, the driver returns an error indicating that the session timed out and the attempt to re-establish the session failed.
- **Web service request timeouts**. You can configure the driver to never time out while waiting for a response to a Web service request or to wait for a specified interval before timing out by setting the connection option WSTimeout. For fetch requests only, if the request times out, you can configure driver to retry the request a specified number of times by setting the WSRetry Count. connection option. If all subsequent attempts to retry a request fails, the driver returns an error indicating that the service request timed out and the subsequent requests failed. See "Connection option descriptions" for details on the WS Timeout and WS Retry Count connection options.

See also

WS Timeout on page 122
WS Retry Count on page 121
Connection option descriptions on page 69

Isolation and lock levels supported

The driver does not currently support transactions in Google BigQuery. Therefore, locking and isolation levels are not supported.

Unicode support

The Google BigQuery driver is fully Unicode enabled. On UNIX and Linux platforms, the driver supports both UTF-8 and UTF-16. On Windows platforms, the Google BigQuery driver supports UCS-2/UTF-16 only.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 applications on UNIX and Linux" for related details.

Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.

See also

UTF-16 applications on UNIX and Linux on page 44

Parameter metadata support

The driver supports returning parameter metadata as described in this section.

Insert and Update statements

The driver supports returning parameter metadata for the following forms of Insert and Update statements:

- `INSERT INTO FOO VALUES(?, ?, ?)`
- `INSERT INTO FOO (COL1, COL2, COL3) VALUES(?, ?, ?)`
- `UPDATE FOO SET COL1=?, COL2=?, COL3=? WHERE COL1 operator ? [{AND | OR} COL2 operator ?]`

where:

operator

is any of the following SQL operators:

=, <, >, <=, >=, and <>

.

Select statements

The driver supports returning parameter metadata for Select statements that contain parameters in ANSI SQL 92 entry-level predicates, for example, such as COMPARISON, BETWEEN, IN, LIKE, and EXISTS predicate constructs. Refer to the ANSI SQL reference for detailed syntax.

Parameter metadata can be returned for a Select statement if one of the following conditions is true:

- The statement contains a predicate value expression that can be targeted against the source tables in the associated FROM clause. For example:

  ```
  SELECT * FROM FOO WHERE BAR > ?
  ```

  In this case, the value expression "BAR" can be targeted against the table "FOO" to determine the appropriate metadata for the parameter.

- The statement contains a predicate value expression part that is a nested query. The nested query's metadata must describe a single column. For example:

  ```
  SELECT * FROM FOO WHERE (SELECT X FROM Y    WHERE Z = 1) < ?
  ```

The following Select statements show further examples for which parameter metadata can be returned:

```
SELECT COL1, COL2 FROM FOO WHERE COL1 = ? AND COL2 > ?
SELECT ... WHERE COLNAME = (SELECT COL2 FROM T2 WHERE COL3 = ?)
SELECT ... WHERE COLNAME LIKE ?
SELECT ... WHERE COLNAME BETWEEN ? AND ?
SELECT ... WHERE COLNAME IN (?, ?, ?)
SELECT ... WHERE EXISTS(SELECT ... FROM T2 WHERE COL1 < ?)
```

ANSI SQL 92 entry-level predicates in a WHERE clause containing GROUP BY, HAVING, or ORDER BY statements are supported. For example:

```
SELECT * FROM T1 WHERE COL = ? ORDER BY 1
```

Joins are supported. For example:

```
SELECT * FROM T1,T2 WHERE T1.COL1 = ?
```

Fully qualified names and aliases are supported. For example:

```
SELECT A, B, C, D FROM T1 AS A, T2 AS B WHERE A.A = ? AND B.B = ?
```

4

Using the SQL engine server

Some applications may experience problems loading the JVM required for the SQL engine because the process exceeds the available heap space. If your application experiences problems loading the JVM, you can configure the driver to operate in server mode.

The driver supports the following SQL engine modes:

- **Server mode**: The driver's SQL engine runs in a separate process with its own JVM, instead of trying to load the SQL engine and JVM in the same process used by the driver.
- **Direct mode**: The driver operates with the SQL engine and JVM running in a single process.
- **Auto mode**: The driver attempts to run in server mode. However, if server mode is unavailable, the SQL engine will failover to run in direct mode.

By default:

- **Windows**: The driver operates in server mode by default.
- **Linux**: The driver operates in direct mode by default.

Note: You must be an administrator to start or stop the service, or to configure any settings for the service.

See the following sections for details on configuring the SQL engine server on your platform.

For details, see the following topics:

- Configuring server mode using the Configuration Manager
- Configuring the SQL engine server using Java options
- Configuring Java logging for the SQL engine server

Configuring server mode using the Configuration Manager

In server mode, you must start the SQL engine server before connecting.

Note: The Configuration Manager is currently supported only on Windows platforms.

The following sections describe how to configure, start, and stop the SQL engine server using the Configuration Manager.

1. Open your data source using the Configuration Manager; then, select the **SQL Engine** tab.
2. Set the SQL Engine Mode (SQLEngineMode) connection option to one of the following values:
   - **0 - Auto**: The SQL engine attempts to run in server mode first, but will failover to direct mode if server mode is unavailable.
   - **1 - Server**: The SQL engine runs exclusively in server mode.

   Note: By default, SQL Engine Mode is set to **1 - Server** for Windows and **2 - Direct** for Linux.

   The fields associated with server mode will become editable, and the **Start** button appears.
3. Provide values for the following options:
   - **Server Port Number**: Specifies a valid port on which the SQL engine listens for requests from the driver. The default value depends on your platform:

     32-bit: `19946`

     64-bit: `19945`
   - **JVM Classpath**: Specifies the CLASSPATH for the JVM used by the driver. See "JVM Classpath" for details. The default depends on your platform:

     Windows: `{.;c:\`*`install_dir`*`\java\lib\googlebigquery.jar}`

     Linux: `{.:/home/user1/`*`install_dir`*`/java/lib/googlebigquery.jar}`
   - **JVM Arguments**: A string that contains the arguments that are passed to the JVM that the driver is starting. The location of the JVM must be specified on your PATH. See "JVM Arguments" for details. The default value is:

     `-Xmx1024m`
   - **JVM Path**: Specifies fully qualified path to the JVM executable that you want to use to run the SQL engine server. The path must not contain double quotation marks.
4. Optionally, if connecting through a proxy server, provide values for the following options:
   - **Server Proxy Host**: Specifies the host name of the proxy server used by the SQL engine. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
   - **Server Proxy Port**: Specifies the port needed to connect to the proxy server used by the SQL engine.
   - **Server Proxy User**: Specifies the user name needed to connect to the proxy server used by the SQL engine.

- **Server Proxy Password**: Specifies the password needed to connect to the proxy server used by the SQL engine.

Note: After the initial configuration, in order for changes to the required and optional connection option values to take effect, you must restart the SQL engine server.

5. Click **Save** to save your changes
6. Click **Start** to run the SQL engine service. A message is displayed that indicates whether the SQL engine was able to successfully run.

Stopping the SQL engine server using the Configuration Manager

To stop the SQL engine server:

1. Open the Configuration Manager and select the SQL Engine tab.
2. From the SQL Engine Mode drop-down list, select **0 - Auto** or **1 - Server**.
3. Click **Stop** to stop the service. A message is displayed to confirm that the service stopped.
4. Click **Exit** to close the Configuration Manager.

Configuring the SQL engine server using Java options

Before you begin: In server mode, you must start the SQL engine server before using the driver.

The following sections describe how to configure, start, and stop the SQL engine server on Linux platform.

Note: By default, SQL Engine Mode is set to 1 (Server) for Windows and 2 (Direct) for Linux.

To configure the SQL engine server, specify values for the Java options in the following JVM argument to suit your environment. See the "SQL engine server Java options" table for a description of these options.

```
java -Xmx<heap_size>m -cp "<jvm_classpath>" com.ddtek.jdbc.<driver>.phoenix.sql.Server
       -port <port_number> -Dhttp.proxyHost=<proxy_host> -Dhttp.proxPort=<proxy_port>
        -Dhttp.proxyUser=<proxy_user> -Dhttp.proxyPassword=<proxy_password>
```

For example:

```
java -Xmx1024m -cp "/opt/Progress/DataDirect/ODBC_80_64bit/java/lib/googlebigquery.jar"
      com.ddtek.jdbc.googlebigquery.phoenix.sql.Server -port 19945
      -Dhttp.proxyHost=myhost@mydomain.com -Dhttp.proxPort=12345
      -Dhttp.proxyUser=JohnQPublic -Dhttp.proxyPassword=secret
```

To start the SQL engine service, execute the JVM Argument after configuring the Java options. A confirmation message is returned once the server is online.

Note: After the initial configuration, in order for changes to these connection option values to take effect, you must restart the SQL engine server.

Table 2: SQL engine server Java options

Java Option	Description
Required Java Options	
`-cp`	Specifies the CLASSPATH for the Java Virtual Machine (JVM) used by the driver. The CLASSPATH is the search string the JVM uses to locate the Java jar files the driver needs. The driver's JVM is located on the following path: *`install_dir`*`/java/lib/googlebigquery.jar`
`-port`	Specifies a valid port on which the SQL engine listens for requests from the driver. We recommend specifying one of the following values: • `19946` (32-bit drivers) • `19945` (64-bit drivers)
Optional Java Options	
`-Xmx`	Specifies the maximum memory heap size, in megabytes, for the JVM. The default size is determined by your JVM. We recommend specifying a size no smaller than `1024`. **Note:** Although this option is not required to start the SQL engine server, we highly recommend specifying a value.
`-Dhttp.proxyHost`	Specifies the host name of the proxy server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
`-Dhttp.proxyPort`	Specifies the port number where the proxy server is listening for HTTP and/or HTTPS requests.
`-Dhttp.proxyUser`	Specifies the user name needed to connect to the proxy server.
`-Dhttp.proxyPassword`	Specifies the password needed to connect to the proxy server.

Stopping the SQL engine server

To stop the SQL engine server, choose one of the following:

- Using an application, execute `SHUTDOWN` SQL.
- From a command line, press `Ctrl` + `C`.

A message is returned to confirm that the service is stopped.

Configuring Java logging for the SQL engine server

Java logging for the SQL engine server can be configured using either the JVM or the driver.

For details, refer to "Configuring logging" in the *Progress DataDirect for ODBC Drivers Reference*.

5

Connection option descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name.

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Note: The driver does not support specifying values for the same connection option multiple times in a connection string or DSN. If a value is specified using the same attribute multiple times or using both long and short attributes, the connection may fail or the driver may not behave as intended.

The following tables list the connection string attributes supported by the Google BigQuery driver.

General attributes

The following table summarizes the general attributes that are required to connect to a Google BigQuery instance.

Table 3: General attributes

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`oauth2.0`
Dataset (DS)	None

Attribute (Short Name)	Default
DataSourceName (DSN)	None
Description (n/a)	None
Project (PT)	None

OAuth 2.0 attributes

The following table summarizes attributes that are used for OAuth 2.0 authentication.

Table 4: OAuth 2.0 attributes

Attribute (Short Name)	Default
AccessToken (AT)	None
Auth URI (o2au)	`https://accounts.google.com/o/oauth2/auth`
ClientID (CI)	None
ClientSecret (CS)	None
Enable Login Prompt (elp)	`0` (false)
Redirect URI (o2ru)	None
RefreshToken (RT)	None
Scope (SC)	`https://www.googleapis.com/auth/bigquery`
Token URI (o2tu)	`https://accounts.google.com/o/oauth2/token`

Service account attributes

The following table summarizes attributes that are used for service account authentication.

Table 5: Service account attributes

Attribute (Short Name)	Default
JWTAudience (JWTA)	`https://accounts.google.com/o/oauth2/token`
ServiceAccountEmail (SAE)	None
ServiceAccountKeyContent (SAKC)	None

Attribute (Short Name)	Default
ServiceAccountPrivateKey (SAPK)	None
Token URI (o2tu)	`https://accounts.google.com/o/oauth2/token`

Mapping attributes

The following table summarizes attributes involved in mapping the remote Google BigQuery data model to a local schema map used to support SQL queries against Google BigQuery.

Table 6: Mapping attributes

Attribute (Short Name)	Default
ConfigOptions (CFGO)	`VarcharLength=65535; SchemaSet=;`
CreateMap (CM)	2 (NotExist)
SchemaMap (SMP)	Default value depends on environment

Storage API attributes

The following table summarizes attributes that are used to leverage the Google BigQuery Storage API when fetching large result sets.

Table 7: Storage API attributes

Attribute (Short Name)	Default
StorageAPIMinPageCount (SAMPC)	3 (pages)
StorageAPIThreshold (SAT)	`10000` (pages)
UseStorageAPI (USA)	`false`

Legacy SQL attributes

The following table summarizes attributes that can be used to execute queries using legacy SQL.

Table 8: Legacy SQL attributes

Attribute (Short Name)	Default
AllowLargeResults (ALR)	`false`
LegacyDataset (LD)	`_queries_`
LegacyTable (LT)	`_sql_*`
Syntax (SY)	`standard`

Proxy server attributes

The following table summarizes proxy server attributes.

Table 9: Proxy server attributes

Attribute (Short Name)	Default
ProxyHost (PXHN)	Empty string
ProxyPassword (PXPW)	Empty string
ProxyPort (PXPT)	Empty string
ProxyUser (PXUN)	Empty string

Web service attributes

The following table summarizes web service attributes, including those related to timeouts.

Table 10: Web service attributes

Attribute (Short Name)	Default
WSFetchSize (WSFS)	`1000000`
WSPoolSize (WSPS)	`1`
WSRetryCount (WSRC)	`5`
WSTimeout (WST)	`120` (seconds)

Timeout attributes

The following table summarizes timeout attributes.

Table 11: Timeout attributes

Attribute (Short Name)	Default
JobTimeout (JT)	`0` (no timeout)
LoginTimeout (LT)	`15`
WSRetryCount (WSRC)	`5`
WSTimeout (WST)	`120` (seconds)

Pooling attributes

The following table summarizes pooling attributes.

Table 12: Statement pooling attributes

Attribute (Short Name)	Default
Pooling (POOL)	`0`
ConnectionReset (CR)	`0`
LoadBalanceTimeout (LBT)	`0`
MaxPoolSize (MXPS)	`100`
MinPoolSize (MNPS)	`0`

Additional attributes

The following table summarizes additional attributes.

Table 13: Additional attributes

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	`1`
BinaryLength (BL)	`65535`
EnableCatalogSupport (ECS)	`0`
FetchSize (FS)	`100` (rows)
Host Name on page 89	`www.googleapis.com`
IANAAppCodePage (IACP) UNIX ONLY	`4` (ISO 8559-1 Latin-1)
JsonFormat (JF)	`raw`

JVMArgs (JVMA)	For the 32-bit driver when the SQL Engine Mode is set to 2 (Direct):
	`-Xmx256m`
	For all other configurations:
	`-Xmx1024m`

JVMClasspath (JVMC)	*`install_dir`*`\java\lib\googlebigquery.jar`
KMSKeyName (KKN)	None
Location (LN)	None
LogConfigFile (LCF)	Empty string
MaximumBytesBilled (MBB)	`0` (no limit)

Attribute (Short Name)	Default
MaximumBilling Tier (MBT)	0
PrimaryKeyPattern (PKP)	None
RefreshSchema (RS)	1
RefreshSchemaForDDL (RSFD)	true
ReportCodepageConversionErrors (RCCE)	0(Ignore Errors)
RetryExceptions (REX)	0(False)
ServerPortNumber (SPN)	32-bit: 19946 64-bit: 19945
SQLEngineMode (SEM)	For Windows: 1 (Server) For UNIX/Linux: 2 (Direct)
UseStreamingInsert (USI)	1
UseQueryCache (UQC)	1

For details, see the following topics:

- Access Token
- Allow Large ResultSet
- Application Using Threads
- Authentication Method
- Auth URI
- Binary Length
- Client ID
- Client Secret
- Config Options
- Connection Pooling
- Connection Reset
- Create Map
- Dataset
- Data Source Name
- Description

- Enable Login Prompt
- Enable Catalog Support
- Fetch Size
- Host Name
- IANAAppCodePage
- Job Timeout
- Json Format
- JVM Arguments
- JVM Classpath
- JWT Audience
- KMSKeyName
- Legacy Dataset
- Legacy Table
- Location
- LoadBalance Timeout
- Log Config File
- Login Timeout
- MaximumBytesBilled
- MaximumBilling Tier
- Max Pool Size
- Min Pool Size
- Primary Key Pattern
- Project
- Proxy Host
- Proxy Password
- Proxy Port
- Proxy User
- Redirect URI
- Refresh Schema
- Refresh Schema For DDL
- Refresh Token
- Report Codepage Conversion Errors
- Retry Exceptions

Access Token

Attribute

AccessToken (AT)

Purpose

Specifies the access token required to authenticate to a Google BigQuery instance when OAuth 2.0 is enabled (`AuthenticationMethod=oauth2.0`).

The access token acts as a session ID for the connection. See "Generating access token and refresh token" to know how to obtain an access token.

Valid Values

`string`

where:

`string`

is the access token you have obtained to authenticate to Google BigQuery.

Notes

- If a value for the Access Token option is not specified, the driver uses the value of the Refresh Token option to make a connection.
- If both Access Token and Refresh Token values are not specified, the driver cannot make a successful connection.
- If both AccessToken and RefreshToken values are specified, the driver uses the AccessToken value. However, if the AccessToken value expires, it uses the RefreshToken value to generate a new AccessToken value.

Default

None

See also

Configuring OAuth 2.0 authentication on page 50

Generating access tokens and refresh tokens on page 55

Authentication Method on page 78

Refresh Token on page 107

Allow Large ResultSet

Attribute

AllowLargeResultSet (ALR)

Purpose

Determines whether the driver returns results larger than 128 MB for legacy SQL queries.

Note: When AllowLargeResultSet is set to `true`, the results are stored in the dataset and table specified using LegacyDataset and LegacyTable properties.

Valid Values

`false` | `true`

Behavior

If set to `false`, the driver does not support query results larger than 128 MB.

If set to `true`, the driver supports query results larger than 128 MB.

Default

`false`

See also

Legacy Dataset on page 95

Legacy Table on page 96

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

Determines whether the driver works with applications using multiple ODBC threads.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to `0` (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Notes

- This connection option can affect performance.

Default

`1` (Enabled)

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

Determines which authentication method the driver uses when establishing a connection.

Valid Values

`serviceaccount` | `oauth2.0`

Behavior

If set to `oauth2.0`, the driver uses OAuth 2.0 authentication when establishing a connection.

If set to `serviceaccount`, the driver uses service account authentication when establishing a connection.

Default

`oauth2.0`

See Also

Authentication on page 50

Auth URI

Attribute

AuthURI (o2au)

Purpose

Specifies the endpoint for obtaining an authorization code from a private endpoint for OAuth 2.0 implementations.

Valid Values

string

where:

string

is the endpoint for retrieving the OAuth 2.0 authorization code from the third party authorization service.

Notes

When this endpoint is queried, the authorization service presents an interface prompting the user to approve or deny access to backend data .

See "Configuring OAuth 2.0 authentication" for examples and more information.

Default Value

`https://accounts.google.com/o/oauth2/auth`

See also

Configuring OAuth 2.0 authentication on page 50

Binary Length

Attribute

BinaryLength (BL)

Purpose

Specifies the maximum length of characters that the driver supports for BINARY data type columns.

Valid Values

x

where:

x

is a positive integer between 1 and 2147483647.

Behavior

If set to *x*, the driver restricts the characters in the column to the specified value.

Default

65535

Client ID

Attribute

ClientID (CI)

Purpose

Specifies the consumer key for your application. The driver uses this value when authenticating to a Google BigQuery instance using OAuth 2.0 (`AuthenticationMethod=oauth2.0`).

See "Generating access token and refresh token" to know how to obtain the client ID for your application.

Valid Values

string

where:

string

is the consumer key for your application.

Default

None

See also

Configuring OAuth 2.0 authentication on page 50

Generating access tokens and refresh tokens on page 55

Authentication Method on page 78

Client Secret

Attribute

ClientSecret (CS)

Purpose

Specifies the consumer secret for your application. The driver uses this value when authenticating to a Google BigQuery instance using OAuth 2.0 (`AuthenticationMethod=oauth2.0`).

See "Generating access token and refresh token" to know how to obtain the client secret for your application.

Valid Values

`string`

where:

`string`

is the consumer secret for your application.

Default

None

See Also

Configuring OAuth 2.0 authentication on page 50

Generating access tokens and refresh tokens on page 55

Authentication Method on page 78

Config Options

Attribute

ConfigOptions (CFGO)

Purpose

Determines how the mapping of the remote Google BigQuery data model to a local schema map can be configured, customized, and updated.

Notes

- This option is primarily used for initial configuration of the driver for a particular user. It is not intended for use with every connection. By default, the driver configures itself and this option is normally not needed. If Config Options is specified on a connection after the initial configuration, the values specified for Config Options must match the values specified for the initial configuration.

Valid Values

`key` = *`value`*`;[`*`key`* = *`value`*`;]`

where:

`key`

is one of the following configuration options:

- `VarcharLength`

- SchemaSet

The value is a set of key value pairs separated by a semicolon (;). The value must be enclosed in curly brackets. For example:

```
VarcharLength=65535;SchemaSet=;
```

Default

```
VarcharLength=65535;SchemaSet=;
```

See also

SchemaSet (Config Option) on page 82

VarcharLength (Config Option) on page 83

SchemaSet (Config Option)

Attribute

SchemaSet (SS)

Purpose

Specifies the project-dataset pairs that you want the driver to fetch metadata for.

Valid Values

String | *

where:

String

is a comma-separated list of project and dataset pairs (*project*:*dataset*).

Behavior

If set to *String*, the driver fetches metadata for the specified project and dataset pairs. For example, if you set SchemaSet=project1:dataset1,project1:dataset2, the driver fetches metadata for only dataset1 and dataset2 of project1. If you want the driver to fetch metadata for all the datasets of a project, put an asterisk (*) after the colon. For example, SchemaSet=project1:*.

If set to *, the driver fetches metadata for all the projects and datasets your account has access to.

Notes

- If you do not specify a value for SchemaSet, the driver fetches metadata for the project and dataset specified at the time of connection.
- The driver does not fetch metadata for public projects by default. To fetch metadata for public projects, specify them using the SchemaSet configuration option.

Default

The project and dataset specified at connection.

See also

VarcharLength (Config Option) on page 83

VarcharLength (Config Option)

Attribute

VarcharLength (VL)

Purpose

Specifies the maximum length the driver supports for String type columns.

Valid Values

x

where:

x

is a positive integer between 1 and 2147483647 indicating the maximum length.

Default

65535

See also

SchemaSet (Config Option) on page 82

Connection Pooling

Attribute

Pooling (POOL)

Purpose

Specifies whether to use the driver's connection pooling.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Notes

- The application must be thread-enabled to use connection pooling.

- This connection option can affect performance.

Default

0 (Disabled)

Connection Reset

Attribute

ConnectionReset (CR)

Purpose

Determines whether the state of connections that are removed from the connection pool for reuse by the application is reset to the initial configuration of the connection.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the state of connections removed from the connection pool for reuse by an application is reset to the initial configuration of the connection. Resetting the state can negatively impact performance because additional commands must be sent over the network to the server to reset the state of the connection.

If set to 0 (Disabled), the state of connections is not reset.

Notes

- This connection option can affect performance.

Default

0 (Disabled)

Create Map

Attribute

CreateMap (CM)

Purpose

Determines whether the driver creates a new schema map when establishing the connection.

Valid Values

0 | 1 | 2 | 3

Behavior

If set to 0 (No), the driver uses the current schema map specified by the Schema Map option. If one does not exist, the connection fails.

If set to 1 (ForceNew), the driver deletes the current schema map specified by the Schema Map option and creates a new one at the same location.

Warning: This causes all views, data caches, and map customizations defined in the current schema map to be lost.

If set to 2 (NotExist), the driver uses the current schema map specified by the Schema Map option. If one does not exist, the driver creates one.

If set to 3 (Session), the driver does not create or use schema map. It stores metadata in the memory.

Default

2 (NotExist)

See also

- Schema Map on page 109

Dataset

Attribute

Dataset (DS)

Purpose

Specifies the name of the dataset that you want the driver to connect to. The datasets in Google BigQuery are equivalent to schemas in ODBC.

Note: If you want to query data in a dataset different from the one you specified at the time of connection, specify it in the following format:

```
project.dataset.table
```

Valid Values

string

where:

string

is the name of your Google BigQuery dataset.

Default

None

See also

Configuring OAuth 2.0 authentication on page 50

Project on page 102

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

Specifies the name of a data source in your Windows Registry or `odbc.ini` file.

Valid Values

`string`

where:

`string`

is the name of a data source.

Default

None

Description

Attribute

Description (n/a)

Purpose

Specifies an optional long description of a data source. This description is not used as a runtime connection attribute, but does appear in the `ODBC.INI` section of the Registry and in the `odbc.ini` file.

Valid Values

`string`

where:

`string`

is a description of a data source.

Default

None

Enable Login Prompt

Attribute

EnableLoginPrompt (elp)

Purpose

Specifies whether the driver fetches access and refresh tokens at connection when OAuth2.0 is enabled (`AuthenticationMethod=OAuth2`). When this option is enabled, the driver launches the login prompt for your service at connection, which allows you to specify your login credentials and initiate the dynamic authorization code grant flow. Enabling this option provides a method of fetching access and refresh tokens without using the Configuration Manager or a third-party tool.

Valid Values

`0` | `1`

Behavior

If set to `0` (false), the driver does not launch the login prompt for your service. If access and refresh tokens are needed for your authentication flow, you will need to fetch them using the Configuration Manager or a third-party tool.

If set to `1` (true), the driver opens the login prompt for your service when attempting to connect. Submitting user and password credentials via the prompt initiates the dynamic authorization code grant to fetch access and refresh tokens.

Notes

- This option is used only for the dynamic authorization code grant flow. See "Dynamic authorization code grant" for a full list of requirements.
- This option is supported only when the SQL Engine is running in direct mode (`SQLEngineMode=2`).
- When this option is enabled, the value of the Redirect URI (`RedirectURI`) option must include the port number. For example, `RedirectURI=http://localhost:80` or `RedirectURI=http://localhost:8080`.

Default Value

`0` (false)

See Also

- Dynamic authorization code grant on page 51

Enable Catalog Support

Attribute

EnableCatalogSupport (ECS)

Purpose
Determines whether the driver supports specifying values for catalog parameters in metadata calls. Note that catalogs and schemas are equivalent to projects and datasets in Google BigQuery.

Valid Values
`1 | 0`

Behavior
If set to `1`, a value can be specified for the catalog parameter in metadata calls. For example: `SQLTables("MyProject","Dataset1","Employee",Null)`, where `MyProject` is a catalog, `Dataset1` is a schema, and `Employee` is a table.

If set to `0`, no value can be specified for the catalog parameter in metadata calls. The values for catalog and schema must be specified within the schema parameter, seperated by a period. For example: `SQLTables(Null,"MyProject.Dataset1","Employee",Null)`, where `MyProject` is a catalog, `Dataset1` is a schema, and `Employee` is a table.

Notes
- The driver can fetch metadata only for:
  - The project and dataset the application is connected to.
  - The project and dataset specified using the SchemaSet config option.
- When the value for the Enable Catalog Support connection option is changed, the SchemaMap file must be refreshed.

Default
`0`

Fetch Size

Attribute
FetchSize (FS)

Purpose
Specifies the maximum number of rows that the driver processes before returning data to the application. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes improve overall fetch times at the cost of additional memory.

FetchSize is related to, but different than, WSFetchSize. WSFetchSize specifies the number of rows of raw data that the driver fetches from the remote data source, while FetchSize specifies how many of these raw data rows the driver processes before returning data to the application. Processing the data includes converting from the remote data source data type to the driver SQL data type used by the application. If FetchSize is greater than WSFetchSize, the driver makes multiple round trips to the data source to get the requested number of rows before returning control to the application.

Valid Values
`0 | `*`x`*

where:

x

is a positive integer.

Behavior

If set to `0`, the driver fetches and processes all of the rows of the result before returning control to the application.

If set to *x*, the driver fetches and processes the specified number of rows before returning data to the application.

Notes

- WSFetchSize and FetchSize can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.

Default

`100`

See Also

- WS Fetch Size on page 119

Host Name

Attribute

HostName (HOST)

Purpose

Specifies the host name portion of the Google BigQuery API endpoint to which you send requests.

Valid Values

`url`

where:

`url`

is the host name portion of the Google BigQuery API endpoint to which you send requests. For example: `www.googleapis.com`.

Notes

- The ServerName attribute is an alias for the Host Name (HostName) option.

Default

`www.googleapis.com`

IANAAppCodePage

Supported on UNIX, Linux, and macOS only.

Attribute

IANAAppCodePage (IACP)

Purpose

An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode enabled or if your database character set is not Unicode.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (`odbc.ini`)
- In the ODBC section of the system information file (`odbc.ini`)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of `4` (ISO 8859-1 Latin-1).

Valid Values

`IANA_code_page`

where:

`IANA_code_page`

is one of the valid values listed in "IANAAppCodePage values" in the *Progress DataDirect for ODBC Drivers Reference*. The value must match the database character encoding and the system locale.

Default

`4` (ISO 8559-1 Latin-1)

See Also

Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.

Job Timeout

Attribute

JobTimeout (JT)

Purpose

Specifies the time, in seconds, that the driver waits for a job to run before timing it out.

Valid Values

`0` | *x*

where:

x

is a positive integer that defines the number of seconds the driver waits for a job to run.

Behavior

If set to `0`, the driver waits indefinitely for a job to run; there is no timeout.

If set to *x*, the driver uses the value as the default timeout for any job run against a Google BigQuery instance.

Default

`0` (no timeout)

Json Format

Attribute

JsonFormat (JF)

Purpose

Determines the JSON string format in which the driver returns values for complex data types, such as Array and Struct.

Valid Values

`raw` | `keyvalue` | `pretty` | `unsafe`

Behavior

If set to `raw`, the values are returned in their native Google BigQuery format.

If set to `keyvalue`, the values are returned in key value pairs. Also, if there is a closing curly bracket (`}`) or a back slash (`\`) in a value, the driver escapes it by adding a back slash (`\`) in front of it. For example, if the value is `"8}"`, the driver returns it as `"8\}"`.

If set to `pretty`, only the values are returned (unaccompanied by keys).

If set to `unsafe`, the values are returned in key value pairs. However, if there are any special characters in them, they are not escaped.

Examples

If the data type is Simple Array and values are [121,122,123], the driver returns the values in one of the following formats based on the valid value you set for the Json Format option:

Valid Value	Data Format
keyvalue	[{v=121},{v=122},{v=123}]
pretty	[121, 122, 123]
raw	[{"v":"121"},{"v":"122"},{"v":"123"}]
unsafe	[{v=121},{v=122},{v=123}]

Default

raw

JVM Arguments

Attribute

JVMArgs (JVMA)

Purpose

A string that contains the arguments that are passed to the JVM that the driver is starting. The location of the JVM must be specified on the driver library path. For information on setting the location of the JVM in your environment, see:

- Setting the library path environment variable (Windows) on page 45
- Library search path on page 36 (UNIX/Linux)

When specifying the heap size for the JVM, note that the JVM tries to allocate the heap memory as a single contiguous range of addresses in the application's memory address space. If the application's address space is fragmented so that there is no contiguous range of addresses big enough for the amount of memory specified for the JVM, the driver fails to load, because the JVM cannot allocate its heap. This situation is typically encountered only with 32-bit applications, which have a much smaller application address space. If you encounter problems with loading the driver in an application, try reducing the amount of memory requested for the JVM heap. If possible, switch to a 64-bit version of the application.

Valid Values

string

where:

string

contains arguments that are defined by the JVM. Values that include special characters or spaces must be enclosed in curly braces { } when used in a connection string.

Example

To set the heap size used by the JVM to 256 MB and the http proxy information, specify:

```
{-Xmx256m -Dhttp.proxyHost=johndoe -Dhttp.proxyPort=808}
```

To set the heap size to 256 MB and configure the JVM for remote debugging, specify:

```
{-Xmx256m
-Xrunjdwp:transport=dt_socket, address=9003,server=y,suspend=n -Xdebug}
```

Default

For the 32-bit driver when the SQL Engine Mode connection option is set to 2 (Direct):

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

JVM Classpath

Attribute

JVMClasspath (JVMC)

Purpose

Specifies the CLASSPATH for the Java Virtual Machine (JVM) used by the driver. The CLASSPATH is the search string the JVM uses to locate the Java jar files the driver needs.

Valid Values

string

where:

string

specifies the CLASSPATH. Separate multiple jar files by a semi-colon on Windows platforms and by a colon on Linux and UNIX platforms. CLASSPATH values with multiple jar files must be enclosed in curly braces { } when used in a connection string.

If your process employs multiple drivers that use a JVM, the value of the JVM Classpath for all affected drivers must include an absolute path to all the jar files for drivers used in that process. In addition, the value specified must be identical for all drivers. For example, if you are using the Google BigQuery and MongoDB drivers on Windows, you would specify a value of {c:*install_dir*\java\lib\googlebigquery.jar; c:*install_dir*\java\lib\mongodb.jar} for both drivers. If the value for any of the affected drivers is missing a file path or is different from the one specified for the other drivers, the drivers will return an error at connection that the JVM is already running.

Example

On Windows:

```
{.;c:\install_dir\java\lib\googlebigquery.jar}
```

On UNIX:

```
{.:/home/user1/install_dir/java/lib/googlebigquery.jar}
```

Default

install_dir\java\lib\googlebigquery.jar

JWT Audience

Attribute

JWTAudience (JWTA)

Purpose

Specifies the JWT audience claim associated with your service account. It is required to authenticate to Google BigQuery when the Service Account authentication is enabled (`AuthenticationMethod=serviceaccount`). To learn more about service accounts and JWT audience claims, refer to the Google documentation.

Valid Values

String

where:

String

is the JWT audience claim associated with your service account.

Default Value

`https://accounts.google.com/o/oauth2/token`

See also

Configuring service account authentication on page 53

KMSKeyName

Attribute

KMSKeyName (KKN)

Purpose

Specifies the customer-managed encryption key (CMEK) that the driver uses for executing queries. If it is not specified, the driver uses the default key encryption key from Google.

To learn more about CMEK, refer to the Google documentation.

Valid Values

projects/*project*/locations/*location*/KeyRings/*keyring*/cryptoKeys/*key*

where:

project

specifies the name of the project that you want the driver to connect to.

location

specifies the geographical location where your dataset is stored.

keyring

specifies the key ring value, which is a prerequisite for creating CMEK. To learn how to create a key ring, refer to the Google documentation.

key

specifies the CMEK value. To learn how to create a key, refer to the Google documentation.

Notes

- Passing KMSKeyName as part of job configuration is not supported for DDL statements. Therefore, for Create statements, CMEK must be provided using the Options clause, in the following format:

  ```
  CREATE TABLE <dataset_name>.<table_name>(<column_name> <column_type>)
  OPTIONS(kms_key_name='projects/project/locations/location/KeyRings/keyring/cryptoKeys/key')
  ```

- If a table is encrypted using CMEK, you can perform insert and select operations on it with or without specifying CMEK. However, you must not specify an incorrect CMEK, as it leads to query failure.
- If you specify a CMEK to query a table that is not ecrypted with CMEK, the query will fail.
- CMEKs specified at connection are used to execute queries for the life of the connection.

Default

None

Legacy Dataset

Attribute

LegacyDataset (LD)

Purpose

Specifies the dataset where results for legacy SQL queries are stored when Allow Large ResultSet is enabled (`AllowLargeResultSet=true`).

Valid Values

`string`

where:

`string`

is the name of the dataset where results for legacy SQL queries are stored when Allow Large ResultSet is enabled (`AllowLargeResultSet=true`).

Default

`_queries_`

See also

Allow Large ResultSet on page 77

Legacy Table on page 96

Legacy Table

Attribute

LegacyTable (LT)

Purpose

Specifies the table where results for legacy SQL queries are stored when Allow Large ResultSet is enabled (`AllowLargeResultSet=true`).

Valid Values

`string`

where:

`string`

is the name of the table where results for legacy SQL queries are stored when Allow Large ResultSet is enabled (`AllowLargeResultSet=true`).

Default

`_sql_*`

See also

Allow Large ResultSet on page 77

Legacy Dataset on page 95

Location

Attribute

Location (LN)

Purpose

Specifies the geographical location where your dataset is stored. Google BigQuery allows storing datasets in either one single geographical place, such as Tokyo, or a large geographical area, such as Europe.

For more information on dataset locations, refer to the Google BigQuery documentation.

Valid Values

string

where:

string

is the geographical location where your dataset is stored.

Default

None

See also

Dataset on page 85

LoadBalance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

Specifies the number of seconds to keep inactive connections open in a connection pool. An inactive connection is a database session that is not associated with an ODBC connection handle, that is, a connection in the pool that is not in use by an application.

Valid Values

0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to *0*, inactive connections are kept open.

If set to *x*, inactive connections are closed after the specified number of seconds passes.

Notes

- The Min Pool Size option may cause some connections to ignore this value.
- This connection option can affect performance.

Default

0

Log Config File

Attribute

LogConfigFile (LCF)

Purpose

Specifies the filename of the configuration file used to initialize the driver logging mechanism.

If the driver cannot locate the specified file when establishing the connection, the connection fails and the driver returns an error.

Valid Values

string

where:

string

is the relative or fully qualified path of the configuration file used to initialize the driver logging mechanism. If the specified file does not exist, the driver continues searching for an appropriate configuration file.

Default

Empty string

See Also

For more information, refer to "Logging for Java components" in the *Progress DataDirect for ODBC Drivers Reference*.

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

The number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error. To override the value that is set by this connection option for an individual connection, set a different value in the SQL_ATTR_LOGIN_TIMEOUT connection attribute using the SQLSetConnectAttr() function.

Valid Values

`0` | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to `0`, the connection request does not time out, but the driver responds to the `SQL_ATTR_LOGIN_TIMEOUT` attribute.

If set to *x*, the connection request times out after the specified number of seconds unless the application overrides this setting with the `SQL_ATTR_LOGIN_TIMEOUT` attribute.

Default

`0`

MaximumBytesBilled

Attribute

MaximumBytesBilled (MBB)

Purpose

Specifies the maximum number of bytes a query can read. As per the on-demand pricing model of Google BigQuery, charges are billed based on the number of bytes a query reads. To control the cost a query may incur, you can set a limit. Once the limit is exceeded, the query fails without incurring any cost.

Valid Values

`0` | *x*

where:

x

is a positive integer that defines the maximum number of bytes a query can read.

Behavior

If set to `0`, the driver allows queries to read indefinite amount of data; there is no limit.

If set to *x*, the driver uses the value as the limit beyond which queries fail without incurring any cost.

Default

`0` (no limit)

MaximumBilling Tier

Attribute

MaximumBillingTier (MBT)

Purpose

Specifies the billing tier that you have access to. If you query a resource beyond the limit set for your tier, the query will fail without incurring any cost.

Valid Values

x

where:

x

is a positive integer that identifies the tier you have access to.

Default

`0`

Max Pool Size

Attribute

MaxPoolSize (MXPS)

Purpose

The maximum number of connections allowed within a single connection pool. When the maximum number of connections is reached, no additional connections can be created in the connection pool.

Valid Values

An integer from `1` to `65535`

For example, if set to 20, the maximum number of connections allowed in the pool is 20.

Notes

This connection option can affect performance.

Default

100

Min Pool Size

Attribute

MinPoolSize (MNPS)

Purpose

Specifies the minimum number of connections that are opened and placed in a connection pool, in addition to the active connection, when the pool is created. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

Valid Values

0 | *x*

where:

x

is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

Behavior

If set to 0, no connections are opened in addition to the current existing connection.

Notes

- This connection option can affect performance.

Default

0

Primary Key Pattern

Attribute

PrimaryKeyPattern (pkp)

Purpose

Determines which column in a table is designated as the primary key in the metadata returned by the driver. Google BigQuery does not have the concept of primary keys, or even uniqueness. However, some applications will not function properly without at least one column in a table designated as the primary key. This option allows your applications that require a primary key to function correctly when connecting to Google BigQuery data sources.

Valid Values

** | column_name*

where

column_name

is the name of the column, specified as a regular expression, that you want designated as the primary key in each table.

Behavior

If set to *, the driver designates the first column in each table that is not of the BOOL, RECORD, ARRAY or GEOGRAPHY data types as the primary key.

If set to column_name, the driver designates the primary key as the first column in each table whose name matches the specified regular expression. The driver will not designate any column that is of the BOOL, RECORD, ARRAY, or GEOGRAPHY data types as the primary key.

If no value is specified, the driver does not designate a primary key for any tables. This is the typical behavior for Google BigQuery data sources.

Default Value

No default value

Project

Attribute

Project (PT)

Purpose

Specifies the name of the project that you want the driver to connect to. The projects in Google BigQuery are equivalent to catalogs in ODBC.

Note: If you want to query data in a project different from the one you specified at the time of connection, specify it in the following format:

project.dataset.table

Valid Values

string

where:

string

is the name of your Google BigQuery project.

Default

None

GUI Tab

String

See also

Configuring OAuth 2.0 authentication on page 50

Dataset on page 85

Proxy Host

Attribute

ProxyHost (PXHN)

Purpose

Specifies the Hostname of the Proxy Server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the server or a fully qualified domain name to which you want to connect.

The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.

Default

Empty string

Proxy Password

Attribute

ProxyPassword (PXPW)

Purpose

Specifies the password needed to connect to the Proxy Server.

Valid Values

String

where:

String

specifies the password to use to connect to the Proxy Server. Contact your system administrator to obtain your password.

Default

Empty string

Proxy Port

Attribute

ProxyPort (PXPT)

Purpose

Specifies the port number where the Proxy Server is listening for HTTP and/or HTTPS requests.

Valid Values

port_name

where:

port_name

is the port number of the server listener. Check with your system administrator for the correct number.

Default

None

Proxy User

Attribute

ProxyUser

Purpose

Specifies the user name needed to connect to the Proxy Server.

Valid Values

The default user ID that is used to connect to the Proxy Server.

Default

Empty string

Redirect URI

Attribute

RedirectURI (o2ru)

Purpose

Specifies the endpoint to which the client is returned after third-party authorization for OAuth 2.0 implementations.

Valid Values

String

where:

String

is the endpoint to which the client is returned after third-party authorization. For example, `http://localhost:8080`.

Notes

- The redirect URI is often registered with the authentication service to provide improved security. Registering the endpoint prevents your valid authentication credentials being redirected to a malicious site; therefore, reducing the risk of sharing your access token and other sensitive information with unauthorized parties.
- See "OAuth 2.0 authentication" for examples and more information.

Default Value

No default value

Refresh Schema

Attribute

RefreshSchema (RS)

Purpose

Specifies whether the driver automatically refreshes the map of the data model when a user connects to a REST service.

Valid Values

`1 | 0`

Behavior

If set to `1` (Enabled), the driver automatically refreshes the map of the data model when a user connects to a REST service. Changes to objects since the last time the map was generated will be shown in the metadata.

If set to `0` (Disabled), the driver does not refresh the map of the data model when a user connects to a REST service.

Notes

- This option should not be enabled (`RefreshSchema=1`) when `CreateMap=session`.

Default

`1`

Refresh Schema For DDL

Attribute

RefreshSchemaForDDL (RSFD)

Purpose

Determines whether the driver automatically refreshes the map of the data model when a user performs a DDL operation (Create or Drop).

Valid Values

`true | false`

Behavior

If set to `true`, the driver automatically refreshes the map of the data model when a user performs a DDL operation. As a result, the DDL operations take longer than usual to complete. Also, changes to objects since the last time the map was generated will be shown in the metadata.

If set to `false`, the driver does not refresh the map of the data model when a user performs a DDL operation. As a result, the DDL operations take less time to complete. However, the metadata calls may return inaccurate results because changes to objects since the last time the map was generated will not be shown in the metadata.

Notes

- This property should not be enabled (`RefreshSchemaForDDL=true`) when `CreateMap=session`.

Default

`true`

Refresh Token

Attribute

RefreshToken (RT)

Purpose

Specifies the refresh token used to either request a new access token or renew an expired access token. If an access token is not provided or expires at the time of connection, the access token generated using refresh token is used to authenticate to a Google BigQuery instance when OAuth 2.0 is enabled (`AuthenticationMethod=oauth2.0`).

See "Generating access token and refresh token" to know how to obtain a refresh token.

Valid Values

`string`

where:

`string`

is the refresh token you have obtained to authenticate to Google BigQuery.

Notes

- If a value for the Access Token option is not specified, the driver uses the value of the Refresh Token option to make a connection.
- If both Access Token and Refresh Token values are not specified, the driver cannot make a successful connection.
- If both AccessToken and RefreshToken values are specified, the driver uses the AccessToken value. However, if the AccessToken value has expired, it uses the RefreshToken value to generate a new AccessToken value.

Default

None

See also

Configuring OAuth 2.0 authentication on page 50

Generating access tokens and refresh tokens on page 55

Authentication Method on page 78

Access Token on page 76

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another.

An error message or warning can occur if an ODBC call causes a conversion error, or if an error occurs during code page conversions to and from the database or to and from the application. The error or warning generated is `Code page conversion error encountered`. In the case of parameter data conversion errors, the driver adds the following sentence: `Error in parameter` *x*, where *x* is the parameter number. The standard rules for returning specific row and column errors for bulk operations apply.

Valid Values

`0` | `1` | `2`

Behavior

If set to `0` (Ignore Errors), the driver substitutes 0x1A for each character that cannot be converted and does not return a warning or error.

If set to `1` (Return Error), the driver returns an error instead of substituting 0x1A for unconverted characters.

If set to `2` (Return Warning), the driver substitutes 0x1A for each character that cannot be converted and returns a warning.

Default

`0` (Ignore Errors)

Retry Exceptions

Attribute

RetryExceptions (rex)

Purpose

Determines whether the driver retries an API call execution if an HTTP failure or driver exception occurs. The number of retry attempts is specified by the value of the WS Retry Count (WSRetryCount) connection option. By default, exceptions caused during a call execution cannot be retried.

Valid Values

`0` | `1`

Behavior

If set to `1` (true), the driver uses the value specified by the connection option WSRetryCount to retry the API call when an exception occurs.

If set to `0` (false), the driver does not retry the API call when an exception occurs.

Default

`0` (false)

See also

WS Retry Count on page 121

Schema Map

Attribute

SchemaMap (SMP)

Purpose

Specifies either the name or the absolute path and name of the configuration file where the relational map of native data is written. The driver looks for this file when connecting to a Google BigQuery instance. If the file does not exist, the driver creates one.

Valid Values

`string`

where:

`string`

is either the name or the absolute path and name (including the `.config` extension) of the configuration file. For example, if Schema Map is set to a value of:

- `ABC`, the driver either creates or looks for the configuration file `ABC` in the working directory of your application.
- `C:\Users\Default\AppData\Local\Progress\DataDirect\GoogleBigQuery_Schema\jsmith@defcorp.config`, the driver either creates or looks for the configuration file `jsmith@defcorp.config` in the directory `C:\Users\Default\AppData\Local\Progress\DataDirect\GoogleBigQuery_Schema`.

Notes

- When connecting to a Google BigQuery instance, the driver looks for the schema map configuration file. If the configuration file does not exist, the driver creates a schema map using the name and location you have provided. If you do not provide a name and location, the driver creates one using default values.
- The driver uses the path specified in this connection option to store additional internal files.
- You can refresh the internal files related to an existing view of your data by using the SQL extension Refresh Map. Refresh Map runs a discovery against your native data and updates your internal files accordingly.

Default

The default is determined by the environment. The driver attempts to create the files in a subdirectory of the first available directory in the following order:

- Windows
  - DD_HOME environment variable
  - LOCALAPPDATA environment variable
  - APPDATA environment variable
  - User's home directory

  For Windows, the file path takes the following format:

  available_location\Progress\DataDirect\GoogleBigQuery_Schema*user_name*.config

- UNIX/Linux
  - DD_HOME environment variable
  - User's home directory

  For UNIX/Linux, the file path takes the following format:

  available_location/progress/datadirect/GoogleBigQuery_schema/*user_name*.config

See Also

- Configuring OAuth 2.0 authentication on page 50

Scope

Attribute

Scope (SC)

Purpose

Specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token at the time of connection.

Valid Values

string

where:

string

is a space-separated list of security scopes.

Examples

The following example demonstrates a configuration that allows the user to view and manage tables created from Google drive.

```
Scope=https://www.googleapis.com/auth/drive
```

Default

`https://www.googleapis.com/auth/bigquery`

See Also

- Access Token on page 76
- Configuring OAuth 2.0 authentication on page 50

Server Port Number

Attribute

ServerPortNumber (SPN)

Purpose

Specifies a valid port on which the SQL engine listens for requests from the driver.

Valid Values

port_name

where:

port_name

is the port number of the server listener. Check with your system administrator for the correct number.

Notes

- This option is ignored when SQL Engine Mode is set to 2 (Direct).

Default

For the 32-bit driver:

`19946`

For the 64-bit driver:

`19945`

Service Account Email

Attribute

ServiceAccountEmail (SAE)

Purpose

Specifies the email address associated with your service account that is required to authenticate to a Google BigQuery instance when service account authentication is enabled (`AuthenticationMethod=serviceaccount`).

To learn more about service accounts and service account emails, refer to Google documentation.

Valid Values

`string`

where:

`string`

is your service account's email address.

Default

None

See also

Configuring service account authentication on page 53

Service Account Private Key on page 113

Service Account Key Content

Attribute

ServiceAccountKeyContent (SAKC)

Purpose

Specifies the private key required to authenticate to a Google BigQuery instance when service account authentication is enabled (`AuthenticationMethod=serviceaccount`). The private key is obtained from the private key file. The private key file can be downloaded from Google Cloud Platform (GCP) Console.

To learn more about service accounts and service account private keys, refer to Google documentation.

Valid Values

`string`

where:

string

is the value of the `private_key` property in the private key file downloaded from the Google Cloud Platform (GCP) Console. Surrounding quotation marks should be omitted when specifying the value.

Notes

- The value of the Service Account Key Content option should be protected for security reasons. The option has been intentionally excluded from the Windows Setup dialog. In addition, the value should not be hardcoded in an ODBC data source. Rather, it may be specified in the application with the SQLConnect ODBC API, in a connection string as with SQLDriverConnect and SQLBrowseConnect, or through the Logon dialog prompt. (If your operating system supports the Logon dialog, the prompt appears when the value has not been otherwise specified.)
- Either Service Account Key Content or Service Account Private Key may be used to configure service account authentication. Service Account Key Content specifies the private key itself, whereas Service Account Private Key specifies the full path to the `.json` or `.p12` file that contains the private key. When Service Account Key Content is used, the specified private key is used to authenticate the user with the database. When Service Account Private Key is used, the driver extracts the private key value from the specified file and uses it to authenticate the user to the database. If you do not want to persist the private key file in your environment, you should use the Service Account Key Content option. If Service Account Key Content and Service Account Private Key are both specified, the Service Account Key Content option will be used to connect to Google BigQuery.

Example

When specifying the value of Service Account Key Content as with this connection string example, surrounding quotation marks should be omitted.

```
ServiceAccountKeyContent=NJJXZexIHJFGYBgkqhkiG9w0BAQnWRwiHANpf3MC1pVRqhtTE5tSpxZeQnICG
4zp087Eidn4qc66udg8KAHknyqFdj7b\n+MgxMFPAvJ59cylHFaHA4pGmeGfVqzYub6LEs9aN/751jmZqcuAYp
5nXRF1EvJPN\nsDuJGLvuuDBZW0iux0liEHmcQVBBKwIx8t+EQxePGTiLsBoCdzOUsi4UWWv\nASqfdP/kSX+N
```

Default

None

GUI Tab

None

See also

Configuring service account authentication on page 53

Service Account Email on page 112

Service Account Private Key on page 113

Service Account Private Key

Attribute

ServiceAccountPrivateKey (SAPK)

Purpose

Specifies the full path to the `.json` or `.p12` private key file that contains the key required to authenticate to a Google BigQuery instance when service account authentication is enabled (`AuthenticationMethod=serviceaccount`). You can dowload the private key file from Google Cloud Platform (GCP) Console.

To learn more about service accounts and service account private keys, refer to Google documentation.

Valid Values

`string`

where:

`string`

is the full path to the `.json` or `.p12` private key file.

Notes

- Either Service Account Key Content or Service Account Private Key may be used to configure service account authentication. Service Account Key Content specifies the private key itself, whereas Service Account Private Key specifies the full path to the `.json` or `.p12` file that contains the private key. When Service Account Key Content is used, the specified private key is used to authenticate the user with the database. When Service Account Private Key is used, the driver extracts the private key value from the specified file and uses it to authenticate the user to the database. If you do not want to persist the private key file in your environment, you should use the Service Account Key Content option. If Service Account Key Content and Service Account Private Key are both specified, the Service Account Key Content option will be used to connect to Google BigQuery.

Default

None

See also

Configuring service account authentication on page 53

Service Account Email on page 112

Service Account Key Content on page 112

SQL Engine Mode

Attribute

SQLEngineMode (SEM)

Purpose

Specifies whether the driver's SQL engine runs in the same 32-bit process as the driver (direct mode) or runs in a process that is separate from the driver (server mode). You must be an administrator to modify the server mode configuration values, and to start or stop the SQL engine service.

Valid Values

`0` | `1` | `2`

Behavior

If set to `0` (Auto), the SQL engine attempts to run in server mode first; however, if server mode is unavailable, it runs in direct mode. To use server mode with this value, you must start the SQL engine service before using the driver (see "Starting the SQL engine server" for more information).

If set to `1` (Server), the SQL engine runs in server mode. The SQL engine operates in a separate process from the driver within its own JVM. You must start the SQL Engine service before using the driver (see "Starting the SQL engine server" for more information).

If set to `2` (Direct), the SQL engine runs in direct mode. The driver and its SQL engine run in a single process within the same JVM.

Important: Changes you make to the server mode configuration affect all DSNs sharing the service.

Default

For Windows:

`1` (Server)

For UNIX/Linux:

`2` (Direct)

See Also

- Using the SQL engine server on page 63

Storage API Min Page Count

Attribute

StorageAPIMinPageCount (SAMPC)

Purpose

Specifies a number of pages that, if exceeded, signals the driver to use the Google BigQuery Storage API for select operations. For this behavior to take effect, the Use Storage API option must be set to `true` (enabled), and the value of the Storage API Threshold option must be exceeded.

Valid Values

x

where:

x

is a positive integer that indicates a number of pages in a result set.

Default

3 (pages)

See also

- Storage API attributes
- Use Storage API on page 118
- Storage API Threshold on page 116

Storage API Threshold

Attribute

StorageAPIThreshold (SAT)

Purpose

Specifies a number of rows that, if exceeded, signals the driver to use the Google BigQuery Storage API for select operations. For this behavior to take effect, the Use Storage API option must be set to `true` (enabled), and the value of the Storage API Min Page Count option must be exceeded.

Valid Values

x

where:

x

is a positive integer that indicates a number of rows in a result set.

Default

`10000` (rows)

See also

- Storage API attributes
- Use Storage API on page 118
- Storage API Min Page Count on page 115

Syntax

Attribute

Syntax (SY)

Purpose

Specifies the Google BigQuery SQL dialect to be used for querying data.

Valid Values

`standard` | `legacy`

Behavior

If set to `standard`, the driver uses the standard SQL dialect for querying data.

If set to `legacy`, the driver uses the legacy SQL dialect for querying data.

Default

`standard`

See also

Standard and legacy SQL support on page 22

Token URI

Attribute

TokenURI (o2tu)

Purpose

Specifies the endpoint for retrieving access tokens when OAuth 2.0 or Service Account authentication methods are enabled.

Valid Values

`String`

where:

`String`

is the endpoint used to retrieve access tokens.

Default Value

`https://accounts.google.com/o/oauth2/token`

See also

Configuring OAuth 2.0 authentication on page 50

Configuring service account authentication on page 53

Use Query Cache

Attribute

UseQueryCache (UQC)

Purpose

Determines whether the driver uses Google BigQuery's query cache to save results.

Valid Values

`1` | `0`

Behavior

If set to `1`, the driver uses query cache to save results.

If set to `0`, the driver does not use query cache.

Default

`1`

Use Storage API

Attribute

Use Storage API (USA)

Purpose

Specifies whether the driver uses the Google BigQuery Storage API when fetching large result sets based on the values of the Storage API Threshold and Storage API Min Page Count connection options.

Valid Values

`1` | `0`

Behavior

If set to `1`, the driver uses the Storage API for selects when the number of rows in the result set exceeds the value of the Storage API Threshold option, and the number of pages in the result set exceeds the value of the Storage API Min Page Count option.

If set to `0`, the driver does not use the Storage API, and the Storage API Threshold and Storage API Min Page Count options are ignored.

Default

`0`

See also

- Storage API attributes
- Storage API Min Page Count on page 115
- Storage API Threshold on page 116

Use Streaming Insert

Attribute

UseStreamingInsert (USI)

Purpose

Specifies whether the driver uses the Google BigQuery Streaming API (InsertAll) to execute streaming batch insert operations. This option improves performance for INSERT operations.

Valid Values

`1 | 0`

Behavior

If set to `1` (enabled), the driver uses the streaming API for batch insert.

If set to `0` (disabled), the driver driver executes row by row insert.

Notes

- The driver stops subsequent batch insertion if an invalid row is encountered.
- The Google BigQuery streaming API can be used for INSERT operations only.

Default

`1`

See also

- Google BigQuery Streaming API on page 59
- Performance considerations on page 57

WS Fetch Size

Attribute

WSFetchSize (WSFS)

Purpose

Specifies the number of rows of data the driver attempts to fetch for each ODBC call.

Valid Values

`0` | *x*

where:

x

is a positive integer from `1` to `2147483647` that defines a number of rows.

Behavior

If set to `0`, the driver attempts to fetch up to a maximum of 2147483647 rows. This value typically provides the maximum throughput.

If set to *x*, the driver attempts to fetch up to a maximum of the specified number of rows. Setting the value lower than 1000000 can reduce the response time for returning the initial data. Consider using a smaller WSFetch Size for interactive applications only.

Notes

- WSFetchSize and FetchSize can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.

Default

`1000000`

See also

Fetch Size on page 88

WS Timeout on page 122

WS Pool Size

Attribute

WSPoolSize (WSPS)

Purpose

Specifies the maximum number of Google BigQuery sessions the driver uses. This allows the driver to have multiple web service requests active when multiple ODBC connections are open, thereby improving throughput and performance.

Valid Values

x

where:

x

is the number of Google BigQuery sessions the driver uses to distribute calls. This value should not exceed the number of sessions permitted by your Google BigQuery account.

Notes

- You can improve performance by increasing the number of sessions specified by this option. By increasing the number of sessions the driver uses, you can improve throughput by distributing calls across multiple sessions when multiple connections are active.
- The maximum number of sessions is determined by the setting of WSPoolSize for the connection that initiates the session. For subsequent connections to an active session, the setting is ignored and a warning is returned. To change the maximum number of sessions, close all connections using the Google BigQuery ODBC driver; then, open a new ODBC Google BigQuery connection with desired limit specified for this option.

Default

`1`

WS Retry Count

Attribute

WSRetryCount (WSRC)

Purpose

The number of times the driver retries a timed-out Select, Insert, Update, or Delete request. The timeout period is specified by the WSTimeout (WST) connection option.

Valid Values

`0` | *x*

where:

x

is a positive integer.

Behavior

If set to `0`, the driver does not retry timed-out requests after the initial unsuccessful attempt.

If set to *x*, the driver retries the timed-out request the specified number of times.

Default

`5`

See also

WS Timeout on page 122

WS Timeout

Attribute

WSTimeout (WST)

Purpose

Specifies the time, in seconds, that the driver waits for a response to a Web service request.

Valid Values

`0` | *x*

where:

x

is a positive integer that defines the number of seconds the driver waits for a response to a Web service request.

Behavior

If set to `0`, the driver waits indefinitely for a response; there is no timeout.

If set to *x*, the driver uses the value as the default timeout for any statement created by the connection.

If a Select request times out and WSRetryCount (WSRC) is set to retry timed-out requests, the driver retries the request the specified number of times.

Default

`120` (seconds)

See also

WS Retry Count on page 121