Progress DataDirect for ODBC for Google Analytics 4

User's Guide

Release 8.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Google Analytics 4 Driver

The Progress® DataDirect® for ODBC™ for Google Analytics™ 4 driver supports SQL read-only access for ODBC applications to Google Analytics. The driver supports Google Analytics 4 (GA4) Data and Admin APIs.

Note: The driver does not support the Google Analytics Management API version 3. For Management API support, use The Progress® DataDirect® for JDBC™ for Google Analytics driver.

To support SQL access to Google Analytics services, the driver creates a relational map of the Google Analytics data model and translates SQL statements to Google Analytics REST API requests. In addition, the driver employs a SQL engine component that provides support to SQL constructs unavailable in Google Analytics. This functionality offers a number of advantages, including support for reporting data and metadata in a form that ODBC applications are ready to use.

For an overview of the relational tables exposed by the driver and their corresponding API calls, see Introduction to the Google Analytics 4 data model.

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:

- What's new in this release?
- Driver requirements
- Installing and setting up the driver (Windows)

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 Release

- **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.

Highlights of 8.0.0 Release

- The driver supports SQL read-only access to Google Analytics using Google Analytics 4 (GA4) Data and Admin API. SeeSupported SQL statements and extensions on page 89 for details.
- The driver supports all ODBC Core and Level 1 functions and some Level 1 and Level 2 features.

  Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for additional information.
- The driver supports Google Analytics data types. See Data types on page 18 for details.
- The driver supports OAuth 2.0 authentication. See OAuth 2.0 Authentication on page 44 for further details.
- The driver supports the handling of large result sets with paging, and the Fetch Size (FetchSize) connection option. See Fetch Size on page 70 for details.
- On Windows platforms, the driver includes an enhanced setup dialog, the DataDirect ODBC Driver Configuration Manager, for quick configuration and testing of your driver. This Configuration Manager allows you to:
  - Configure data sources
  - Generate and edit connection strings
  - Test connect data sources and connection strings

- Execute SQL commands for testing
- Add tables to a relational view of Google Analytics data
- Access connection option descriptions and the full product documentation

See Configuring data sources with the Configuration Manager on page 33 and Generating connection strings with the Configuration Manager on page 35 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 43 for details.

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).

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."

4. Select either the **User DSN**, **System DSN**, or **File 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 your driver and click **Finish** to display the driver Setup dialog box.

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

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`.
   - **Host Name**: (Optional) Specify the name or the IP address of the server to which you want to connect. For example: `https://analyticsdata.googleapis.com`.

   For OAuth 2.0 refresh token grant:

   - **Client ID**: Specify the client ID key for your application when authenticating to Google Analytics.
   - **Client Secret**: Specify the client secret for your application when authenticating to Google Analytics.
   - **Refresh Token**: Specify the refresh token used to either request a new access token or renew an expired access token for OAuth 2.0 implementations.

   Note:

   - You can fetch tokens using the Configuration Manager .See OAuth 2.0 Authentication on page 44 for more information.
   - Client ID, Client Secret, and Refresh Token connection options can also be specified in the logon dialog box or passed by your application.
   - The driver supports a number of authentication methods. See OAuth 2.0 Authentication on page 44 for more information.

   ---

6. Define the tables in the relational schema of your Google Analytics data.

   For more information about custom tables, and how they can be used to write useful queries, see Introduction to the Google Analytics Data Model.

   a) On the **Schema Settings** tab, click **Configure Logical Schema**.

   b) Select values for the following drop-down fields:

      - **Account ID**: The account ID used to filter the contents of the table.
      - **Property ID**: The property ID used to filter the contents of the table.

   c) Click the **Create Table** button. In the **Create Table** window, specify the name of table you want to create in the **Table Name** field; then, click **Create**

      Note: If you are editing an existing table that is specified by the **Add Tables** field in the connection string, select the table from the **Select Table** drop-down field.

      ---

   d) Select the metrics and dimensions you want to expose with the table. You may specify up to ten metrics and seven dimensions per table.

   e) Click **Save & Close**.

Result: The **Add Tables** field is populated with a table definition in the form of a JSON string, and the **Default Query Options** field is populated with parameters used for WHERE clause filtering.

7. 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 string examples on page 17 provides connection string examples that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
   - Connection option descriptions on page 57 provides a complete list of supported options by functionality.
   - Configuring data sources with the Configuration Manager on page 33 guides you through using the GUI to configure the driver.
   - Performance considerations on page 49 describes connection options that affect performance, along with recommended settings.

   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.

8. Click **Test Connect** to attempt to connect to the data source using the connection options.
9. The logon dialog appears. If not already specified, update the fields provided; then, click **OK**.

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

10. If the test was successful, the window displays a confirmation message.
11. 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 Configuration 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.
12. Connect to your 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.
    - Power BI: Power BI is a business intelligence software program that allows you to generate analytics and visualized representations of your data.
    - 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.
    - Supported SQL statements and extensions on page 89: This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

See also

Using a connection string on page 36

Installing and setting up the driver (Linux)

Important: To query your data, you must define the tables in your schema using the Add Tables (AddTables) option. Although you can set this value manually, we recommend generating a value for this option using the **Configure Logical Schema** tool in the Configuration Manager. The Configuration Manger is currently supported only on Windows platforms, but you can use an evaluation or licensed copy of the Windows driver to create your Add Tables value for the Linux driver. See Installing and setting up the driver (Windows) on page 12 for details.

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."
      - Sets the library path environment variable for your Linux operating system, `LD_LIBRARY_PATH`, to include the directory containing your JVM's `libjvm.so` file. For details, see "Library search 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 used to authenticate using OAuth 2.0 refresh token grant flow.

```
[ODBC Data Sources]
Google Analytics 4=DataDirect 8.0 Google Analytics 4


[Google Analytics 4]
Driver=<install_dir>/lib/xxgoogleanalytics428.yy
...
Description=My Google Analytics 4 Data Source
...
AddTables='{myTableDefinitionString}'
...
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com
...
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd
...
RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
...
```

  See Configuration through the system information (odbc.ini) file on page 37 for more information.

  Note: The ClientID, ClientSecret, and RefreshToken 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.

- **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, DSN-less connections, for more information. For examples, see Connection string examples.

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 string examples provides connection string examples that can be used to configure common functionality and features. You can modify and combine these examples to create a string that best suites your environment.

     Note: The options and values described in "Connection string examples" apply to all configuration methods.

   - Connection option descriptions provides a complete list of supported options by functionality.
   - Performance Considerations describes connection options that affect performance, along with recommended settings.

5. Connect to your 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 is a command-line tool that allows you to test connect, execute SQL statements, and practice using the ODBC API in environments that do not support GUIs.
- Supported SQL statements and extensions on page 89: This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

See also

ODBCINI on page 31
Library search path on page 30
Connection string examples on page 17

Connection string examples

ODBC provides a method for specifying connection information via a connection string and the SQLDriverConnect API. This section provides examples of connection strings configured to use common features and functionality. You can modify and/or combine these examples to create a connection string for your environment.

In addition to the connection strings for DSN-less connections demonstrated in this section, the driver supports DSN and File DSN connection strings. See "Using a connection string" for syntax and detailed information for supported connection string types.

Note: Although you can still connect to Google Analytics, you cannot query your data without defining the tables in your schema using the Add Tables (AddTables) option. You can generate a value for this option using the **Configure Logical Schema** button in the Configuration Manager and copy it into your string. See Configuring data sources with the Configuration Manager for details.

Note: The options and values described in this section apply to all configuration methods.

- OAuth 2 authentication example
- Proxy server example

OAuth 2.0 authentication

This following strings demonstrate the basic options used to connect using OAuth 2.0 authentication.

Refresh token flow

```
DRIVER=DataDirect 8.0 Google Analytics 4;AddTables='{myTableDefinitionString}';
       ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
       ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
       RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;[attribute=value[;...]]
```

For more information on these options and values, see OAuth 2.0 Authentication on page 44.

Proxy server

This string includes the properties used to connect to a Proxy Server with OAuth 2.0 authentication.

```
DRIVER=DataDirect 8.0 Google Analytics 4;AddTables='{myTableDefinitionString}';
       ProxyHost=pserver;ProxyPassword=secret;ProxyPort=808;ProxyUser=proxy_user;
```

```
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;[attribute=value[;...]]
```

For more information on these options and values, see Proxy server options.

See also

Using a connection string on page 36

Data types

The following table lists native data types supported by the driver and how they are mapped to ODBC data types.

Table 1: Google Analytics 4 Driver Data Types

Google Analytics 4 Driver Data Type	ODBC Data Type
BigInt	SQL_BIGINT
Binary	SQL_BINARY
Bit	SQL_BIT
Boolean	SQL_BIT
Char	SQL_CHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_FLOAT
GUID	SQL_CHAR
Integer	SQL_INTEGER
JSON	SQL_VARCHAR
LongVarBinary	SQL_LONGVARBINARY
LongVarChar	SQL_LONGVARCHAR
NVarChar	SQL_UNICODE_VARCHAR
SmallInt	SQL_SMALLINT
Time	SQL_TYPE_TIME

Google Analytics 4 Driver Data Type	ODBC Data Type
Timestamp	SQL_TYPE_TIMESTAMP
TinyInt	SQL_TINYINT
VarBinary	SQL_VARBINARY
VarChar	SQL_VARCHAR

Driver specifications

This section describes the general functionality supported by the driver.

- **ODBC compliance**: The driver is compliant with the Open Database Connectivity (ODBC) 3.52 specification. The driver is ODBC core compliant and supports some Level 1 and Level 2 features.

  Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported API functions.

  The driver supports only the following Level 2 functions:

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

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

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

- **Isolation and lock levels**: Because transactions are not supported, the driver supports only the isolation level 0 (read uncommitted).

  Refer to "Locking and isolation levels" in the *Progress DataDirect for ODBC Drivers Reference* for details.

- **Connections and statements supported**: The driver supports multiple connections and multiple statements per connection.

Additional information

In addition to the content provided in this guide, the documentation set also contains detailed conceptual and reference information that applies to all the drivers. For more information in these topics, refer the *Progress DataDirect for ODBC Drivers Reference* or use the links below to view some common topics:

- "Code page values" lists supported code page values, along with a description, for the Progress DataDirect for ODBC drivers.
- "ODBC API and scalar functions" lists the ODBC API functions supported by Progress DataDirect for ODBC drivers. In addition, it documents the scalar functions that you use in SQL statements.
- "Internationalization, localization, and Unicode" provides an overview of how internationalization, localization, and Unicode relate to each other. It also includes a background on Unicode, and how it is accommodated by Unicode and non-Unicode ODBC drivers.
- "Security best practices for ODBC applications" describes the security best practices you should employ when developing and deploying your application with the driver.

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.

May 2023, 8.0.0 Release of Progress DataDirect for ODBC for Google Analytics 4, Version 0001

2

Tutorials

The following sections guide you through using the driver to access your data with some common third-party applications. For information on installing your driver and setting the CLASSPATH, see "Installing and setting-up the driver (Windows)" or "Installing and setting-up the driver (Linux)."

For details, see the following topics:

- The Example application
- Power BI (Windows only)
- Tableau (Windows only)
- Microsoft Excel (Windows only)

The Example application

The driver installation includes an ODBC application called Example that can be used to connect to a data source and execute SQL.

1. After you have configured the data source, navigate to the `instal_dir\samples\example` directory.
2. Open the application:
   - On Windows, double-click the `Example.exe` file.
   - On Linux, run the example application.

   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.SYSTEM_TABLES
   ```

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

Note: Refer to the `example.txt` file in the `example` subdirectory for a detailed explanation of how to build and use this application.

Power BI (Windows only)

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

1. Navigate to the `\tools\Power BI` subdirectory of the Progress DataDirect installation directory; then, locate the installation batch file `install.bat`.
2. Run the `install.bat` file. The following operations are executed by running the `install.bat` file:
   - The Power BI connector file, `DataDirectGoogleAnalytics4.pqx`, is copied to the following directory.

     ```
     %USERPROFILE%\Documents\Power BI Desktop\Custom Connectors
     ```

   - The following Windows registry entry is updated.

     ```
     HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Power BI
     Desktop\TrustedCertificateThumbprints
     ```

3. Open the Power BI desktop application.
4. From the **Get Data** window, navigate to **Other > Progress DataDirect Google Analytics 4 Connector**.
5. Click **Connect**. Then, from the **Progress DataDirect Google Analytics 4 Connector** window, provide the following information. Then, click **OK**.
   - **Data Source:** Enter a name for the data source. For example, `Google Analytics 4 ODBC DSN`.
   - **SQL Statement:** If desired, provide a SQL command.
   - **Data Connectivity mode:**
     - Select **Import** to import data to Power BI.
     - Select **DirectQuery** to query live data. (For details, including limitations, refer to the Microsoft Power BI article Use DirectQuery in Power BI Desktop.)
6. Enter authentication information when prompted. Once connected, the **Navigator** window displays schema and table information.
7. Select and load tables. Then, prepare your Power BI dashboard as desired.

You have successfully accessed your data and are now ready to create reports with Power BI. For more information, refer to the Power BI product documentation at Power BI documentation.

Tableau (Windows only)

After you have configured your data source, you can use the driver to access your 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 following Tableau data source file:

   `DataDirect Google Analytics 4.tdc`

2. Copy the Tableau data source file 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 **Other Databases (ODBC)** 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**. The Logon dialog appears pre-populated with the connection information you provided in your data source.
6. If required, type your user name and password; then, click **OK**. The Logon dialog closes. Then, click **Sign in** on the Other Databases (ODBC) 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 schema you want to use. The tables stored in this schema are now available for selection in the Table field.

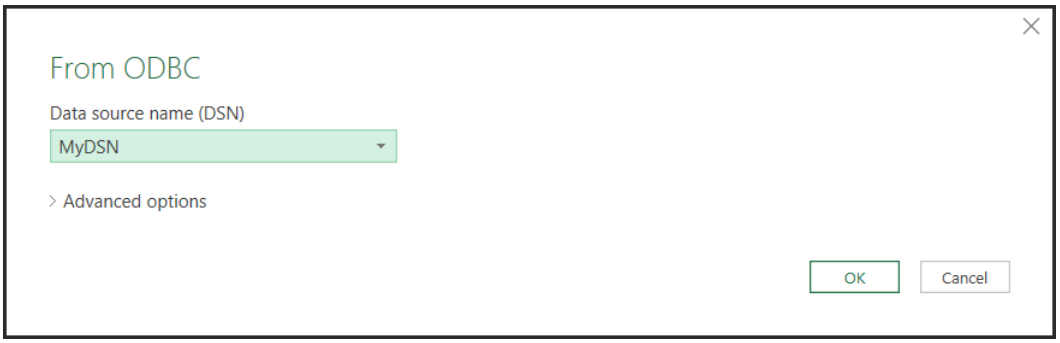
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

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.

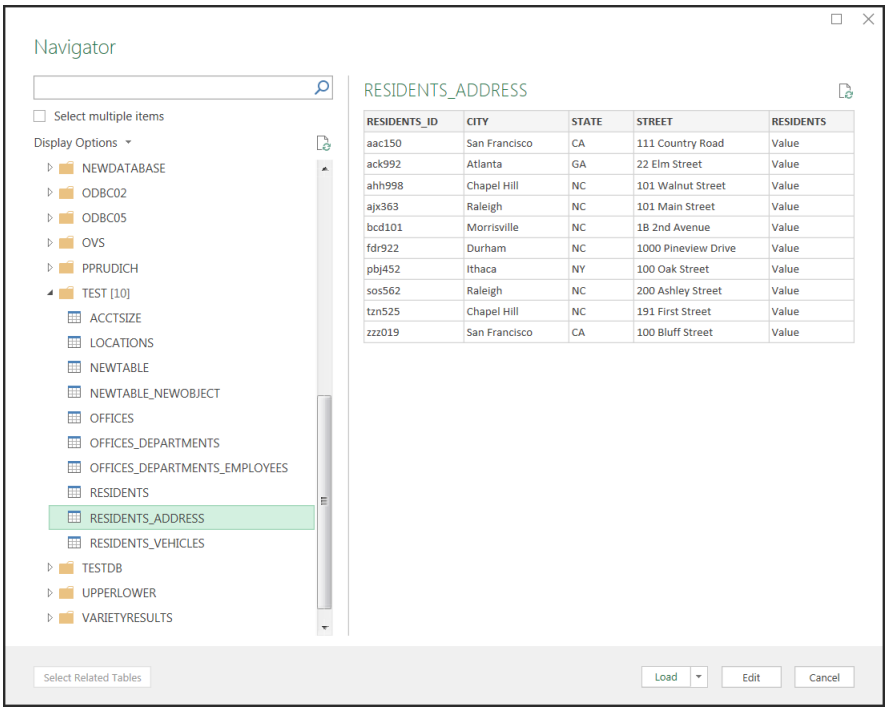
To use the driver to access data with Excel from the Data Connection Wizard:

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.



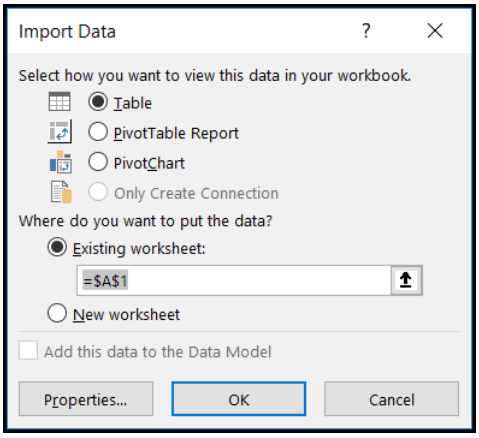
Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

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 options 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 options 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/.

3

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. Data sources store the Information that the driver needs to connect to a database. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on 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 Windows and Linux, 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 "Connection option descriptions" for an alphabetical list of driver connection string attributes and their initial default values.

For details, see the following topics:

- OAuth 2.0 Authentication
- Performance considerations

Environment settings

The first step in setting up and configuring the driver for use is to set environment settings and 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.

Windows environment variables

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: During installation, the Windows installer sets the PATH environment variable to include the path of the JVM.

Linux environment variables

The following topics guide you through setting the environment variables for Linux. You must set these environment variables before connecting with your driver.

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 library search path environment variable, `LD_LIBRARY_PATH`.

The library search path environment variable must be set so that the ODBC core components, Java 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

The product installer places a default system information file, named `odbc.ini`, that contains data sources in the product installation directory. See "Configuration through the system information (odbc.ini) file" 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

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

ODBCINST

The installer program places a default file, named `odbcinst.ini`, for use with DSN-less connections in the product installation directory. 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 40

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 37

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 Linux environment, 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/ivgoogleanylitics4xx.so
```

Note: The full path to the driver does not have to be specified for the tool.

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.

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

UTF-16 applications on 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 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);
    ```

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, Progress DataDirect Google Analytics 4 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 a driver Setup dialog box, 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 Google Analytics 4 Configuration Manager by selecting ODBC Administrator from the Progress DataDirect program group.
2. Select a 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 dialog.

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

   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

     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 driver Setup dialog box.

   - **Note:** Configuring an existing or new file data source using the File DSN tab is not currently supported with the configuration manager.

   The Google Analytics 4 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. Define the tables in the relational schema of your Google Analytics data.

   For more information about custom tables, and how they can be used to write useful queries, see Introduction to the Google Analytics Data Model.

   a) On the **Schema Settings** tab, click **Configure Logical Schema**.

   b) Select values for the following drop-down fields:

      - **Account ID**: The account ID used to filter the contents of the table.
      - **Property ID**: The property ID used to filter the contents of the table.

   c) Click the **Create Table** button. In the **Create Table** window, specify the name of table you want to create in the **Table Name** field; then, click **Create**

      Note: If you are editing an existing table that is specified by the **Add Tables** field in the connection string, select the table from the **Select Table** drop-down field.

   d) Select the metrics and dimensions you want to expose with the table. You may specify up to ten metrics and seven dimensions per table.

   e) Click **Save & Close**.

      Result: The **Add Tables** field is populated with a table definition in the form of a JSON string, and the **Default Query Options** field is populated with parameters used for WHERE clause filtering.

5. 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**.

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

6. Click **Save** to apply your values as the default when connecting with the data source.

See also

Generating connection strings with the Configuration Manager on page 35
Connection option descriptions on page 57
Connection string examples on page 17

Generating connection strings with the Configuration Manager

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

The Progress DataDirect Google Analytics 4 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.

See also

Configuring data sources with the Configuration Manager on page 33
Testing connections and queries with the Configuration Manager on page 42

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 driver for Linux or Windows is:

```
DSN=GoogleAnalytics4;ProxyUser=JOHN;ProxyPassword=XYZZY
```

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

```
FILEDSN=GoogleAnalytics4;ProxyUser=JSMITH;ProxyPassword=XYZZY
```

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

```
DRIVER=DataDirect 8.0 Google Analytics 4;AddTables='{myTableDefinitionString}';
       ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
       ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
       RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

See also

Connection option descriptions on page 57
Sample default odbc.ini file on page 38
File data sources on page 41
DSN-less connections on page 40
Connection option descriptions on page 57
Connection string examples on page 17

Additional configuration methods for Linux

This section contains configuration methods that are specific to the Linux environment.

Configuration through the system information (odbc.ini) file

In the 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.

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.

Note: The driver and driver manager support ASCII and UTF-8 encoding in the `odbc.ini` file.

Refer to the "Character encoding in the odbc.ini and odbcinst.ini files" in *Progress DataDirect for ODBC Drivers Reference* for details.

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

```
Google Analytics 4=DataDirect 8.0 Google Analytics 4
```

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 Analytics 4]`. 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 Linux driver uses if individual data sources have not specified a different value. See "IANAAppCodePage" in the "Connection option descriptions" for details.

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*.

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 Analytics 4=DataDirect 8.0 Google Analytics 4

[Google Analytics 4]
Driver=ODBCHOME/lib/ivgoogleanalytics428.so
Description=DataDirect 8.0 Google Analytics 4 Driver
AccessToken=
AddTables=
ApplicationUsingThreads=1
AuthenticationMethod=24
AuthURI=https://accounts.google.com/o/oauth2/auth
ClientID=
ClientSecret=
DebugRecord=
DefaultQueryOptions=
FetchSize=100
HostName=https://analyticsdata.googleapis.com
InitializationString=
JVMArgs=-Xmx256m
JVMClasspath=
JVMPath=
KeywordConflictSuffix=
LogConfigFile=ddlogging.properties
ProxyHost=
ProxyPassword=
ProxyPort=0
ProxyUser=
RedirectURI=
RefreshToken=
ReportCodepageConversionErrors=0
Scope=https://www.googleapis.com/auth/analytics.manage.users.readonly
      https://www.googleapis.com/auth/analytics.readonly
      https://www.googleapis.com/auth/analytics
ServerPortNumber=19972
```

```
ServerProxyHost=
ServerProxyPassword=
ServerProxyPort=
ServerProxyUser=
SQLEngineMode=0
StmtCallLimit=0
StmtCallLimitBehavior=1
TokenURI=https://accounts.google.com/o/oauth2/token
TransactionMode=0
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 values for attributes in the data source definitions as necessary based on your system specifics.

     See "Connection option descriptions" for other specific attribute values.

  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, `[My Datasource]`.
  d) Modify the attributes in the new definition as necessary based on your system specifics.

     See "Connection option descriptions" for other specific attribute values.

  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" section 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 57
Connection option descriptions on page 57

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.

Note: The driver and driver manager support ASCII and UTF-8 encoding in the odbcinst.ini file.

Refer to the "Character encoding in the odbc.ini and odbcinst.ini files" in *Progress DataDirect for ODBC Drivers Reference* for details.

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

```
DataDirect 8.0 SAP S4HANA=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

Configuration through the system information (odbc.ini) file on page 37
ODBCINST on page 31
Configuration through the system information (odbc.ini) file on page 37

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 Analytics 4=Installed

[DataDirect 8.0 Google Analytics4]
```

```
Driver=ODBCHOME/lib/ivgoogleanalytics428.so
JarFile=ODBCHOME/java/lib/googleanalytics4.jar
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
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 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 or Linux. See "Configuring and connecting to data sources" for a general description of ODBC data sources on both Windows and Linux.

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 Analytics 4
AddTables='{myTableDefinitionString}'
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd
RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831
```

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\GoogleAnalytics4.dsn
```

or, on Linux:

```
FILEDSN=/home/users/john/filedsn/GoogleAnalytics4.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 "Configuration through the system information (odbc.ini) file" 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/GoogleAnalytics4.dsn;
DefaultQueryOptions=startDate=50;RefreshToken=1//11a1=bCD/EfGh1Ijk+Lgd1g-11tk1c1111
```

See also

Configuring and connecting to data sources on page 29
DSN-less connections on page 40
Configuration through the system information (odbc.ini) file on page 37
Configuring and connecting to data sources on page 29
DSN-less connections on page 40
Configuration through the system information (odbc.ini) file on page 37

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 Analytics 4 Configuration Manager.

To test your connection and query:

1. Open the Google Analytics 4 Configuration Manager by selecting the ODBC Administrator from the Progress DataDirect group.

   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 Analytics 4 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**.

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

See also

Configuring data sources with the Configuration Manager on page 33

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:

- 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 the access token used to authenticate to your service.
- In the Refresh Token field, type the refresh token used to connect to your service.
- Optionally, in the Scope field, type a space-separated list of OAuth scopes.

OAuth 2.0 Authentication

The driver supports OAuth 2.0 to access Google Analytics resources. Before you can configure the driver for OAuth, you must register your client application with Google Cloud and obtain information such as the client ID, the client secret, and the refresh token. The following workflow describes the process for setting up OAuth 2.0 access.

1. Register your application with Google Cloud. See this topic for step-by-step instructions for registering your application.
2. Obtain client ID and client secret. If the application has already been registered, see this topic for steps to obtain the client ID and client secret.
3. Obtain access and refresh tokens using the Configuration Manager. To configure the driver, you will need to specify either an access token or a refresh token. This topic provides instructions to obtain these tokens using the driver's Configuration Manager.
4. Configure the driver to use OAuth 2.0. You can configure the driver to access Google Analytics resources by specifying the connection options described in this topic.

Note: For more information, refer to the Using OAuth 2.0 to Access Google APIs section of the *Google Identity Help*.

Registering your application with Google Cloud

Registering your client application with Google Cloud involves three basic steps: creating a Google Project, registering the application, and creating OAuth credentials.

- Create a Google Cloud project
- Register the application
- Create OAuth credentials

Create a Google Cloud project

If you do not already have a Google Cloud project to access Google Analytics resources, you must create one to register a client application.

Take the following steps to create a Google Cloud project.

1. Go to Google Developer Console.
2. Navigate to **Enabled APIs & services**.
3. Create the project using one of the following methods:
   - If you have no projects, click **CREATE PROJECT**.

- If you are in a project but want to create a new one, select the project you are in, and then click **NEW PROJECT**.

4. Specify the project name in the **Project name** field.
5. Specify the location in the **Location** field.
6. Click **CREATE**.

Results:

You have created a Google Cloud project. From within this project, you will register your client application and create OAuth credentials.

Register the application

Take the following steps to register an application to access the resources of a Google Cloud project.

1. From the Google Developer Console, navigate to the **APIs & Services > OAuth Consent screen**.
2. Select the **User Type**.
3. Click **CREATE**.
4. Complete the application registration forms.

   OAuth Consent Screen form

   a. Specify values for the following fields.

   - App name
   - User support email
   - App logo
   - Application home page
   - Application privacy policy link
   - Application terms of service link
   - Authorized domain
   - Developer contact information

   b. Click **SAVE AND CONTINUE**.

   Scopes form

   a. Click **ADD OR REMOVE SCOPES**.

   b. Select the scope or scopes to specify permissions for the client application. In some cases, you may need to manually enter scopes. The following scopes are the default scopes used by the driver and provide read-only access to Google Analytics resources.

   - **View and download data:**

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

     ```
     https://www.googleapis.com/auth/analytics.readonly
     ```

   - **View user permissions:**

     ```
     https://www.googleapis.com/auth/analytics.manage.users.readonly
     ```

Note: The values you select should be saved to a secure location. You will need to specify these values to obtain access and refresh tokens, as described in Obtaining access and refresh tokens using the Configuration Manager.

c. Click **UPDATE**.

d. Click **SAVE AND CONTINUE**.

Test users form

a. Click **ADD USERS**.

b. Enter email address of each test user.

c. Click **ADD**.

d. Click **SAVE AND CONTINUE**.

Summary Page

a. Review the summary.

b. Click **BACK TO DASHBOARD**.

Results:

The OAuth consent screen page for your application is displayed. You have completed the registration process for your application.

Create OAuth credentials

Take the following steps to create client ID and client secret OAuth credentials for your application.

1. From the Google Developer Console, navigate to the **Credentials** screen by clicking **Credentials** on the left.
2. Click **CREATE CREDENTIALS** and select **OAuth client ID**.
3. Select an application type, and enter the requested information.

Note: The **Desktop app** option is a common option for initially setting up and testing the driver from `localhost`. If you are using Progress DataDirect Hybrid Data Pipeline, choose **Web application** even if you have deployed Hybrid Data Pipeline on a local machine.

4. Click **CREATE**.

   Step result: A dialog is displayed with the client ID and secret.

5. Save the application information to a secure location using one of the following methods:

   - Download the JSON file and save it to a secure location. The JSON file includes the client ID and secret as well as summary information about the client application, including useful endpoints.
   - Copy the client ID and secret, and save them to a secure location.

Note: The values for the client ID and secret should be saved to a secure location. You will need to specify these values for the ClientID and ClientSecret connection options.

Results:

You have created OAuth credentials for your client application.

Obtaining the client ID and client secret

Take the following steps to obtain the client ID and secret for an application that has already been registered with Google Cloud.

1. Go to the Google Developer Console.
2. Select the Google Cloud project in which the client application has been registered.
3. Select **Credentials** on the left.
4. Under **OAuth 2.0 Client IDs**, select the client application for which you are retrieving the client ID and secret.
5. The client ID and secret appear on the upper right of the screen. Copy this information to a secure location.

Results:

You have obtained OAuth credentials for your client application.

Obtaining access and refresh tokens using the Configuration Manager

You need the following information before you begin.

- The client ID and client secret for the client application
- The scope or scopes that define permissions for the client application

The Configuration Manager uses the authorization code grant to obtain access and refresh tokens from Google Cloud. The following steps describe how you can use the Configuration Manager to obtain access and refresh tokens. In addition, the Configuration Manager produces a connection URL that you can use in your application.

Note: You must allow popups in your browser to obtain access and refresh tokens with the Configuration Manager.

---

1. .

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

2. - 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 Analytics 4 Configuration Manager will generate a data source and connection string based on the values you specify.

3. Enter values for the required connection options on the **Connection** tab.
   - **Client ID**
   - **Client Secret**
   - **Scope**

     The following scopes are the default scopes used by the driver.

- **View and download data:**
  - `https://www.googleapis.com/auth/analytics`
  - `https://www.googleapis.com/auth/analytics.readonly`
- **View user permissions:**

  `https://www.googleapis.com/auth/analytics.manage.users.readonly`

4. Enter any additional values under the **Connection** tab, or other tabs, as desired.
5. Retrieve access and refresh tokens.
   a. Click **Fetch OAuth Token**.
   b. If prompted, enter Google credentials.
   c. Provide consent to allow the Configuration Manager to retrieve the tokens.
   d. The **Access Token** and **Refresh Token** fields populate with values retrieved from Google Cloud.
6. Click **Test Connect** to verify connectivity and run SQL queries against the service.

Results:

The **Access Token** and **Refresh Token** fields contain access and refresh tokens. You can use these tokens to configure the driver and access Google Analytics resources.

The connection string in the **Connection String** field may be copied and used in your ODBC application to access Google Analytics resources.

Note: Not all the values in the resulting connection string may be required. However, they may be used in your ODBC application. The driver ignores any values that are not required.

See also

Configuring data sources with the Configuration Manager on page 33

Configuring the driver to use OAuth 2.0

Prerequisites:

- Client application registered with Google Cloud
- An access token or refresh token

After you have registered your client application with Google Cloud and obtained the required OAuth information, you may configure the driver to access Google Analytics resources using OAuth 2.0.

To configure the driver, you must provide either the access token or the refresh token.

- Set the Access Token (AccessToken) option to specify the access token you have obtained from Google. Generally, an access token is available for a short period of time and may only be used for a single session.
- Set the Refresh Token (RefreshToken) option to specify the refresh token you have obtained from Google. With a valid refresh token, the driver can obtain a new access token. In this way, a refresh token enables application access to Google for multiple sessions over an extended period of time.

The Access Token and Refresh Token options may both be specified. 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 ignores the Access Token value and uses the Refresh Token value to generate a new Access Token value.

For information on setting up OAuth 2.0 authentication, including obtaining access and refresh tokens, see OAuth 2.0 authentication.

The following examples show the specification of the Access Token and Refresh Token options.

Connection string

```
DRIVER=Datadirect 8.0 Google Analytics 4;AddTables='{myTableDefinitionString}';
AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

odbc.ini file

```
[Google Analytics]
Driver=ODBCHOME/lib/ivgoogleanalytics28.so
...
Description=DataDirect 8.0 Google Analytics
...
AddTables='{myTableDefinitionString}'
...
AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;
...
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
...
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
...
RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
...
```

Performance considerations

Application Using Threads (ApplicationUsingThreads): The driver coordinates concurrent database operations (operations from different threads) by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If your application does not make ODBC calls from different threads, the driver has no reason to coordinate operations. In this case, the ApplicationUsingThreads attribute should be disabled (set to `0`).

Note: If you are using a multi-threaded application, you must enable the Application Using Threads option.

Fetch Size (FetchSize): Reducing the number of round trips on the network to the approximate number of rows being fetched increases performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network.

JVM Arguments (JVMArgs): This connection option can be used to address memory and performance concerns by adjusting the max Java heap size. By increasing the max Java heap size, you increase the amount of data the driver accumulates in memory. This can reduce the likelihood of out-of-memory errors and improve performance by ensuring that result sets fit easily within the JVM's free heap space.

See also

Connection option descriptions on page 57

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. Before starting the SQL engine server, choose a directory to store the local database files using the Schema Map (SchemaMap) option. Make sure that you have the correct permissions to write to this directory.

Note: The Configuration Manager is currently supported only on Windows platforms. To configure the SQL engine on Linux platforms, see "Configuring the SQL engine server using Java options."

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: `19972`

     64-bit: `19971`
   - **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\googleanalytics4.jar}`

     Linux: `{.:/home/user1/`*`install_dir`*`/java/lib/googleanalytics4.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.

See also

Configuring the SQL engine server using Java options on page 53
JVM Classpath on page 73
JVM Arguments on page 72

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 start: In server mode, you must start the SQL engine server before using the driver. Before starting the SQL engine server, verify that you have the correct permissions to write to the directory specified by the Schema Map (SchemaMap) option.

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/googleanalytics4.jar"

        com.ddtek.phoenix.sql.Server -port 19971 -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/googleanalytics4.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: • `19972` (32-bit drivers) • `19971` (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.

Java Option	Description
`-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 connection option descriptions in this section 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 the option topics.

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 following tables of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions 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 provide a summary of supported connection options by functionality, including their attribute names, short names, and default values.

General options

The following table summarizes general connection options that can apply to all connections that use data sources.

Table 3: General options

Attribute (Short Name)	Default
DataSourceName (dsn)	No default value
Description (n/a)	No default value
Hostname (host)	`https://analyticsdata.googleapis.com`

OAuth 2.0 authentication options

The following table summarizes options used for OAuth 2.0 authentication.

Table 4: OAuth 2.0 authentication options

Attribute (Short Name)	Default
AccessToken (atok)	No default value
AuthenticationMethod (am)	24 (OAuth2)
AuthURI (o2au)	`https://accounts.google.com/o/oauth2/auth`
ClientID (cid)	No default value
ClientSecret (clse)	No default value
Hostname (host)	`https://analyticsdata.googleapis.com`
RedirectURI (o2ru)	No default value
RefreshToken (rtok)	No default value

Scope (oas)	`https://www.googleapis.com/auth/analytics.manage.users.readonly`
	`https://www.googleapis.com/auth/analytics.readonly`
	`https://www.googleapis.com/auth/analytics`

TokenURI (o2tu)	`https://accounts.google.com/o/oauth2/token`

Mapping options

The following table summarizes connection options involved in mapping the Google Analytics 4 data model to a SQL model.

Table 5: Mapping options

Attribute (Short Name)	Default
AddTables (addt)	No default value
KeywordConflictSuffix (kcs)	No default value
ShowInternalTables (sit)	`0` (false)

SQL engine options

The following table lists the options used to configure the SQL engine.

Table 6: SQL engine options

Attribute (Short Name)	Default

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

JVMClassPath (jvmcp)	*`install_dir`*`\java\lib\googleanalytics4.jar`
JVMPath (jvmp)	*`install_dir`*`\jre\bin\java.exe`
ServerPortNumber (spn)	32-bit driver: `19972` 64-bit driver: `19971`
ServerProxyHost (sph)	No default value
ServerProxyPassword (spw)	No default value
ServerProxyPort (spp)	No default value
ServerProxyUser (spu)	No default value
SQLEngineMode (sem)	For Windows: `0` (Auto) For Linux: The SQL engine runs in Direct mode by default.
SQLService (ss)	No default value

Proxy server options

The following table summarizes proxy server connection options.

Table 7: Proxy server options

Attribute (Short Name)	Default
ProxyHost (pxhn)	No default value
ProxyPassword (pxpw)	No default value
ProxyPort (pxpt)	`0` which means that the default value is determined by whether the value specified for the Proxy Host (ProxyHost) option is an HTTP or HTTPS URL. For HTTP: `80` For HTTPS: `443`
ProxyUser (pxun)	No default value

Web service options

The following table summarizes Web service connection options, including those related to timeouts.

Table 8: Web service options

Attribute (Short Name)	Default
StmtCallLimit (scl)	`0` (no limit)
StmtCallLimitBehavior (sclb)	`1` (ErrorAlways)
WSRetryCount (wsrc)	`5` (attempts)
WSTimeout (wst)	`120` (seconds)

Additional options

The following table summarizes additional connection options.

Table 9: Additional options

Attribute (Short Name)	Default
ApplicationUsingThreads (aut)	`1` (True)
DebugRecord (dbgrecord)	No default value
DefaultQueryOptions (dqo)	`startDate=30daysAgo;endDate=yesterday;startMinutesAgo=29;endMinutesAgo=0`

Attribute (Short Name)	Default
ExtendedOptions (xo)	No default value
FetchSize (fs)	`100` (rows)
InitializationString (is)	No default value
LogConfigFile (lgcf)	`ddlogging.properties`
ReportCodepageConversionErrors (rcce)	`0` (Ignore Errors)
TransactionMode (tm)	`0` (NoTransactions)

For details, see the following topics:

- Access Token
- Add Tables
- Application Using Threads
- Authentication Method
- Authorization URI
- Client ID
- Client Secret
- Data Source Name
- Debug Folder
- Default Query Options
- Description
- Extended Options
- Fetch Size
- Host Name
- Initialization String
- JVM Arguments
- JVM Classpath
- JVM Path
- Keyword Conflict Suffix
- Log Config File
- Proxy Host

Access Token

Attribute

AccessToken (atok)

Purpose

Specifies the access token used to authenticate to Google Analytics 4 with OAuth 2.0 enabled. Typically, this option is configured by the application; however, in some scenarios, you may need to secure a token using external processes. In those instances, you can also use this option to set the access token manually.

Valid Values

`String`

where:

String

is an access token you have obtained from the authentication service.

Notes

- Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically one hour.
- See "OAuth 2.0 authentication" for examples and more information.

Default Value

No default value

See also

OAuth 2.0 Authentication on page 44

Add Tables

Attribute

AddTables (addt)

Purpose

Specifies a JSON string that defines custom tables, including dimensions and metrics, that appear in the Google Analytics 4 schema. You can define the JSON string using the Configure Logical Schema button in the Configuration Manager under the Schema Settings tab. Up to 9 dimensions and 10 metrics can be added to each table.

Valid Values

String

where:

String

is a JSON string that defines custom tables that appear in the Google Analytics 4 schema. Note that this value must be wrapped in single quotation marks.

Example

```
'{"TestTable":["sessions","totalUsers","_browser","_sessionSource"]}'
```

Notes

- Dimension names specified in the string must include a leading underscore character. For example, `"_browser"`. The driver uses the underscore character to distinguish dimensions from metrics.

Default Value

No default value

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` (true), the driver works with single-threaded and multi-threaded applications.

If set to `0` (false), 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 Value

`1` (true)

Authentication Method

Attribute

AuthenticationMethod (am)

Purpose

Determines which authentication method the driver uses during the course of a session.

OAuth 2.0 is currently the only supported authentication method. Therefore, a value for this option does not need to be specified. However, it does appear on the Connection tab of the Configuration Manager.

Valid Values

`24`

Behavior

If set to `24` (OAuth2), the driver uses OAuth 2.0 to authenticate to Google Analytics endpoints. See "OAuth 2.0 authentication" for details.

Default Value

24 (OAuth2)

See also

OAuth 2.0 Authentication on page 44

Authorization URI

Attribute

AuthURI (o2au)

Purpose

Specifies the endpoint for obtaining an authorization code from a third-party authorization service 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 "OAuth 2.0 authentication" for examples and more information.

Default Value

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

See also

OAuth 2.0 Authentication on page 44

Client ID

Attribute

ClientID (cid)

Purpose

Specifies the client ID key for your application when authenticating to Google Analytics 4 with OAuth 2.0 enabled.

Valid Values

String

where:

String

is the client ID key for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 Authentication on page 44

Client Secret

Attribute

ClientSecret (clse)

Purpose

Specifies the client secret for your application when authenticating to Google Analytics 4 with OAuth 2.0 enabled.

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

Valid Values

String

where:

String

is the client secret for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 Authentication on page 44

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 Value

No default value

Debug Folder

Attribute

DebugRecord (dbgrecord)

Purpose

Specifies the directory where the driver generates debug record files. When a value is specified, the driver records server requests and responses to a set of files stored in this location. These files assist in troubleshooting by providing a method for Technical Support to reproduce and debug issues for REST services that are not publicly accessible.

Important: Debug record files may capture security-related headers, such as auth or token headers. Before sending Technical Support debug files, review the content to remove any confidential information that may have been recorded.

Valid Values

`debug_record_folder`

where:

`debug_record_folder`

is the location of the folder where the debug record files are to be generated. For example, `C:\Temp\MyDebug Folder`.

Note: To specify this value using the GUI, you must click the **Record** box . Next, click **Select**; then, browse to or create the file location.

Notes

- The specified directory must exist.
- You must have write access to the specified directory.
- The contents of the specified directory are deleted every time a connection is established.
- For more information, refer to "Enabling debug record mode" in the *Progress DataDirect for ODBC Drivers Reference*.
- For assistance, contact Technical Support.

Default Value

No default value

Default Query Options

Attribute

DefaultQueryOptions (dqo)

Purpose

Specifies the values of Google Analytics 4 query parameters for WHERE clause filtering during a session. Providing values for these parameters simplifies queries.

Valid Values

(key=value[;key=value])

where:

key

is one of the following query parameters:

- `startDate`: The inclusive starting date for the query with Report API. The default is `30daysago` (thirty days prior to the current date).
- `endDate`: The inclusive ending date for the query with Report API. The default is `yesterday` (the day prior to the current date).
- `startMinutesAgo`: The inclusive start minutes for the query with Realtime Report API. The default is `29` (twenty-nine minutes prior to the current time).
- `endMinutesAgo`: The inclusive end minutes for the query with Realtime report API. The default is `0` (current time).
- `currencyCode`: A currency code in ISO4217 format, such as `AED`, `USD`, `JPY`. If the field is empty, the Report API uses the property's default currency. There is no default.
- `keepEmptyRows`: If set to `false` or unspecified, each row with all metrics equal to 0 will not be returned with Report API. If set to `true`, these rows will be returned if they are not separately removed by a filter. The default is `false`.
- `returnPropertyQuota`: If set to `true`, Report or Realtime API will return the current state of this Analytics Property's quota. The default is `false`.

- `accountId`: A Google Analytics 4 Account ID. An Account ID is needed for analytics admin APIs. There is no default.
- `propertyId`: A Google Analytics 4 Property ID. A Property ID is needed for analytics data APIs. A Property ID can be obtained from your Google Analytics dashboard (**Dashboard>Admin>Property>Property Settings>PROPERTY ID**). There is no default.

Notes

- **Important:** In order for `SELECT * FROM` to work in a query with the Data API, `propertyId` must be specified. If `propertyId` is not specified using DefaultQueryOptions, then it must be set explicitly in the WHERE clause.
- **Important:** In order for `SELECT * FROM` to work in a query with the Admin API, `accountId` must be specified. If `accountId` is not specified using DefaultQueryOptions, then it must be set explicitly in the WHERE clause.
- The syntax for `startDate` and `endDate` values is as follows:
  - A date in the format *YYYY-MM-DD*
  - The word `today` for the current date
  - The word `yesterday` for the day prior to the current date
  - *nn*`daysAgo` where *nn* is a number of days prior to the current date

Default Value

If no value is specified (the default), the driver uses the following values:
`startDate=30daysAgo;endDate=yesterday;startMinutesAgo=29;endMinutesAgo=0`.

Description

Attribute

Description (desc)

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 Value

No default value

Extended Options

Attribute

ExtendedOptions (xo)

Purpose

Specifies a semicolon separated list of connection options and their values. Use this connection option to set the value of undocumented connection options that are provided by Progress DataDirect Technical Support.

Valid Values

option=*value*[;*option*=*value*;...]

where:

option

is a connection option.

value

is the value or setting of the connection option.

Default Value

No default value

Fetch Size

Attribute

FetchSize (fs)

Purpose

Specifies the maximum number of rows that the driver processes before returning data to the application when executing a Select. This value provides a suggestion to the driver as to the number of rows it should internally process before returning control to the application. The driver may fetch fewer rows to conserve memory when processing exceptionally wide rows.

Valid Values

0 | *x*

where:

x

is a positive integer indicating the number of rows that should be processed.

Behavior

If set to `0`, the driver processes all the rows of the result before returning control to the application. When large data sets are being processed, setting Fetch Size to `0` can diminish performance and increase the likelihood of out-of-memory errors.

If set to *x*, the driver limits the number of rows that may be processed for each fetch request before returning control to the application.

Notes

- To optimize throughput and conserve memory, the driver uses an internal algorithm to determine how many rows should be processed based on the width of rows in the result set. Therefore, the driver may process fewer rows than specified by Fetch Size when the result set contains exceptionally wide rows. Alternatively, the driver processes the number of rows specified by Fetch Size when the result set contains rows of unexceptional width.
- Fetch Size 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.
- You can use Fetch Size to reduce demands on memory and decrease the likelihood of out-of-memory errors. Simply, decrease Fetch Size to reduce the number of rows the driver is required to process before returning data to the application.

Default Value

`100`

Host Name

Attribute

HostName (host)

Purpose

Specifies the host name portion of the HTTP endpoint to which you send requests.

Valid Values

`url`

where:

`url`

is the host name portion of the HTTP endpoint to which you send requests.

Notes

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

Default Value

`https://analyticsdata.googleapis.com`

Initialization String

Attribute

InitializationString (is)

Purpose

Specifies one or multiple SQL commands to be executed by the driver after it has established a connection and has performed all initialization for the connection. If the execution of a SQL command fails, the connection attempt also fails and the driver throws an exception indicating which SQL command or commands failed.

Valid Values

command[[;*command*]...]

where:

command

is a SQL command.

Notes

Multiple commands must be separated by semicolons. In addition, if this option is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified.

Default Value

No default value

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:

- "Windows environment variables"
- "Library search path" (Linux)

When specifying the heap size for the JVM, 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 Value

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

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

See also

Windows environment variables on page 30
Library search path on page 30
SQL Engine Mode on page 83

JVM Classpath

Attribute

JVMClassPath (jvmcp)

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 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 Analytics 4 and MongoDB drivers on Windows, you would specify a value of {c:\install_dir\java\lib\googleanalytics4.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\googleanalytics4.jar}
```

On Linux:

```
{.:/home/user1/install_dir/java/lib/googleanalytics4.jar}
```

Default Value

No default value

JVM Path

Attribute

JVMPath (jvmp)

Purpose

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.

Valid Values

String

where:

String

The full path to the JVM executable.

Default Value

install_dir\jre\bin\java.exe

Keyword Conflict Suffix

Attribute

KeywordConflictSuffix (kcs)

Purpose

Specifies a string of up to 5 alphanumeric characters that the driver appends to any object or field name that conflicts with a SQL engine keyword.

Valid Values

String

where:

String

is a string of up to 5 alphanumeric characters.

Example

A field called CASE exists in the data schema. To avoid a naming conflict with the SQL engine keyword CASE, you could set KeywordConflictSuffix=TAB. In this scenario, the driver maps the CASE field to the CASETAB column.

Default Value

No default value

Log Config File

Attribute

LogConfigFile (lgcf)

Purpose

Specifies the file name, and optionally, the path of the properties file used to initialize driver logging.

Valid Values

String

where:

String

is the relative or fully qualified path of the properties file to load to initialize driver logging. If you do not specify a path, the driver looks for this file in the current working directory. If the specified file does not exist, the driver continues searching for an appropriate properties file as described in "Logging for Java components" in the *Progress DataDirect for ODBC Drivers Reference*.

Default Value

ddlogging.properties

Proxy Host

Attribute

ProxyHost (pxhn)

Purpose

Identifies a proxy server to use for the first connection.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the proxy server, which may be qualified with the domain name.

IP_address

is an IP address, specified in either IPv4 or IPv6 format, or a combination of the two.

Default Value

No default value

Proxy Password

Attribute

ProxyPassword (pxpw)

Purpose

Specifies the password needed to connect to a proxy server for the first connection.

Valid Values

password

where:

password

is a valid password for that server. Contact your system administrator to obtain a valid password.

Default Value

No default value

Proxy Port

Attribute

ProxyPort (pxpt)

Purpose

Specifies the port number where the proxy server is listening for HTTP or HTTPS requests for the first connection.

Valid Values

port

where:

port

is the port number on which the proxy server is listening. Contact your system administrator to obtain the correct port.

Default Value

0 which means that the default value is determined by whether the value specified for the Proxy Host (ProxyHost) option is an HTTP or HTTPS URL.

For HTTP: 80

For HTTPS: 443

See also

Proxy Host on page 76

Proxy User

Attribute

ProxyUser (pxun)

Purpose

Specifies the user name needed to connect to a proxy server for the first connection.

Valid Values

user_name

where:

user_name

is a valid user ID for the proxy server.

Default Value

No default value

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`.

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

See also

OAuth 2.0 Authentication on page 44

Refresh Token

Attribute

RefreshToken (rtok)

Purpose

Specifies the refresh token used to either request a new access token or renew an expired access token for OAuth 2.0 implementations.

Important: The refresh token is a confidential value used to authenticate to the instance. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the refresh token you have obtained from the authentication service.

Notes

- See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 Authentication on page 44

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 instance 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` (IgnoreErrors), the driver substitutes `0x1A` for each character that cannot be converted and does not return a warning or error.

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

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

Default Value

`0`

Scope

Attribute

Scope (oas)

Purpose

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

Valid Values

String

where:

String

is a space-separated list of security scopes.

Default Value

```
https://www.googleapis.com/auth/analytics.manage.users.readonly
https://www.googleapis.com/auth/analytics.readonly
https://www.googleapis.com/auth/analytics
```

See also

OAuth 2.0 Authentication on page 44

Server Port Number

Attribute

ServerPortNumber (sport)

Purpose

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

Valid Values

port_number

where:

port_number

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 (SQLEngineMode) is set to 2 (Direct).

Default Value

32-bit driver: 19972

64-bit driver: 19971

See also

SQL Engine Mode on page 83

Server Proxy Host

Attribute

ServerProxyHost (sph)

Purpose

Specifies the host name and possibly the domain of the proxy server used by the SQL engine 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 proxy server or a fully qualified domain name to which you want to connect.

IP_address

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

Default Value

No default value

Server Proxy Password

Attribute

ServerProxyPassword (spw)

Purpose

Specifies the password needed to connect to the proxy server used by the SQL engine 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 Value

No default value

Server Proxy Port

Attribute

ServerProxyPort (spp)

Purpose

Specifies the port number of the server listener for the proxy server used by the SQL engine server.

Valid Values

port_name

where:

port_name

is the port number of the server listener of the proxy server used by the SQL engine server. Check with your system administrator for the correct number.

Default Value

No default value

Server Proxy User

Attribute

ServerProxyUser (spu)

Purpose

Specifies the user name needed to connect to the proxy server used by the SQL engine server.

Valid Values

string

where:

string

Is the default user ID that is used to connect to the proxy server used by the SQL engine server.

Default Value

No default value

Show Internal Tables

Attribute

ShowInternalTables (sit)

Purpose

Determines whether the driver exposes the Google Analytics 4 Data table. The Data table is an internal Google Analytics 4 table that contains all the analytics data the has been collected for a website.

Valid Values

`0` | `1`

Behavior

If set to `1` (true), the driver exposes the Data table.

If set to `0` (false), the driver does not expose the Data table.

Default Value

`0` (false)

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 "Using 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 "Using 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 Value

For Windows:

1 (Server)

For Linux:

2 (Direct)

See also

Using the SQL engine server on page 51

SQL Service

Attribute

SQLService (ss)

Purpose

Displays the name of the ODBC SQL engine service that runs as a separate process instead of being loaded within the process of an ODBC application.

Note: This option is used only for display purposes in the configuration manager. No value should be specified for this option.

Default Value

No default value

Statement Call Limit

Attribute

StmtCallLimit (scl)

Purpose

Specifies the maximum number of web service calls the driver can make when executing any single SQL statement or metadata query.

Valid Values

`0` | *x*

where:

x is a positive integer that defines the maximum number of web service calls up to 2147483647 the driver can make when executing any single SQL statement or metadata query.

Behavior

If set to `0`, there is no limit.

If set to *x*, the driver uses this value to set the maximum number of web service calls on a single connection that can be made when executing a SQL statement. This limit can be overridden by changing the `STMT_CALL_LIMIT` session attribute using the `ALTER SESSION` statement. For example, the following statement sets the statement call limit to 10 web service calls:

```
ALTER SESSION SET STMT_CALL_LIMIT=10
```

If the web service call limit is exceeded, the behavior of the driver depends on the value specified for the Statement Call Limit Behavior (StmtCallLimitBehavior) option.

Default Value

`0`

See also

Statement Call Limit Behavior on page 85

Statement Call Limit Behavior

Attribute

StmtCallLimitBehavior (sclb)

Purpose

Specifies the behavior of the driver when the maximum web service call limit specified by the Statement Call Limit (StmtCallLimit) option is exceeded.

Valid Values

`0` | `1`

Behavior

If set to `0` (ReturnResults), the driver returns any partial results it received prior to the call limit being exceeded. The driver generates a warning that not all of the results were fetched.

If set to `1` (ErrorAlways), the driver generates an exception if the maximum web service call limit is exceeded.

Default Value

`1` (ErrorAlways)

See also

Statement Call Limit on page 84

Token URI

Attribute

TokenURI (o2tu)

Purpose

Specifies the endpoint for retrieving access tokens when OAuth 2.0 authentication is enabled.

Valid Values

String

where:

String

is the endpoint used to retrieve access tokens.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

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

See also

OAuth 2.0 Authentication on page 44

Transaction Mode

Attribute

TransactionMode (tm)

Purpose

Specifies how the driver handles manual transactions.

Valid Values

`0` | `1`

Behavior

If set to `0` (NoTransactions), the data source and the driver do not support transactions. Metadata indicates that the driver does not support transactions.

If set to `1` (Ignore), the data source does not support transactions and the driver always operates in auto-commit mode. Calls to set the driver to manual commit mode and to commit transactions are ignored. Calls to rollback a transaction cause the driver to throw an exception indicating that no transaction is started. Metadata indicates that the driver supports transactions and the ReadUncommitted transaction isolation level.

Default Value

`0` (NoTransactions)

Web Service Retry Count

Attribute

WSRetryCount (wsrc)

Purpose

Specifies the number of times the driver retries a timed-out Select request. The timeout period is specified by the Web Service Timeout (WSTimeout) 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 Value

5

See also

Web Service Timeout on page 87

Web Service 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, measured in seconds, for any statement created by the connection.

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

Default Value

`120`

6

Supported SQL statements and extensions

The driver provides support for the SQL statements and the SQL extensions described in this section. SQL extensions are denoted by an (EXT) in the topic title.

For details, see the following topics:

- Alter Session (EXT)
- Explain Plan
- Select
- Subqueries
- SQL expressions

Alter Session (EXT)

Purpose

Changes various attributes of a local or remote session. A local session maintains the state of the overall connection. A remote session maintains the state that pertains to a particular remote data source connection.

Syntax

`ALTER SESSION SET` *`attribute_name=value`*

where:

attribute_name

specifies the name of the attribute to be changed. Attributes apply to either local or remote sessions.

value

specifies the value for that attribute.

The following table lists the local and remote session attributes, and provides descriptions of each.

Table 10: Alter Session Attributes

Attribute Name	Session Type	Description
Current_Schema	Local	Sets the current schema for the local session. The current schema is the schema used when an identifier in a SQL statement is unqualified. The string value must be the name of a schema visible in the local session. For example: `ALTER SESSION SET CURRENT_SCHEMA=GOOGLE ANALYTICS 4`
Stmt_Call_Limit	Local	Sets the maximum number of Web service calls the driver can make in executing a statement. Setting the Stmt_Call_Limit attribute has the same effect as setting the Statement Call Limit connection option. It sets the default Web service call limit used by any statement on the connection. Executing this command on a statement overrides the previously set Statement Call Limit for the connection. The value specified must be a positive integer or `0`. The value `0` means that no call limit exists. For example: `ALTER SESSION SET STMT_CALL_LIMIT=150`

Ws_Call_Count	Remote	Resets the Web service call count of a remote session to the value specified. The value must be `0` or a positive integer. WS_Call_Count represents the total number of Web service calls made to the remote data source instance for the current session. For example:
		`ALTER SESSION SET google analytics 4.WS_CALL_COUNT=0`
		The current value of WS_Call_Count can be obtained by referring to the System_Remote_Sessions system table (see SYSTEM_REMOTE_SESSIONS Catalog Table for details). For example:
		`SELECT * from information_schema.system_remote_sessions WHERE session_id = cursessionid()`

See also

Statement Call Limit on page 84

Explain Plan

Purpose

Retrieves a detailed list of the elements in the execution plan. It generates a result set with a single column named `OPERATION`. The individual elements that comprise the plan are returned as rows in the result set.

Syntax

```
EXPLAIN PLAN FOR {SELECT ... | DELETE ... | INSERT ... | UPDATE ...}
```

The returned list of elements includes the indexes used for performing the query and can be used to optimize the query.

Select

Purpose

Use the Select statement to fetch results from one or more tables.

Syntax

```
SELECT select_clause from_clause
[where_clause]
[groupby_clause]
[having_clause]
[{UNION [ALL | DISTINCT] |
 {MINUS [DISTINCT] | EXCEPT [DISTINCT]} |
 INTERSECT [DISTINCT]} select_statement]
[limit_clause]
```

where:

select_clause

specifies the columns from which results are to be returned by the query. See "Select clause" for a complete explanation.

from_clause

specifies one or more tables on which the other clauses in the query operate. See "From clause" for a complete explanation.

where_clause

is optional and restricts the results that are returned by the query. See "Where clause" for a complete explanation.

groupby_clause

is optional and allows query results to be aggregated in terms of groups. See "Group By clause" for a complete explanation.

having_clause

is optional and specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). See "Having clause" for a complete explanation.

UNION

is an optional operator that combines the results of the left and right Select statements into a single result. See "Union operator" for a complete explanation.

INTERSECT

is an optional operator that returns a single result by keeping any distinct values from the results of the left and right Select statements. See "Intersect operator" for a complete explanation.

EXCEPT | MINUS

are synonymous optional operators that returns a single result by taking the results of the left Select statement and removing the results of the right Select statement. See "Except and Minus operators" for a complete explanation.

orderby_clause

is optional and sorts the results that are returned by the query. See "Order By clause" for a complete explanation.

limit_clause

is optional and places an upper bound on the number of rows returned in the result. See "Limit clause" for a complete explanation.

See also

Select clause on page 92
From clause on page 94
Where clause on page 96
Group By clause on page 97
Having clause on page 97
Union operator on page 98
Intersect operator on page 99
Except and Minus operators on page 100
Order By clause on page 100
Limit clause on page 101

Select clause

Purpose

Use the Select clause to specify with a list of column expressions that identify columns of values that you want to retrieve or an asterisk (*) to retrieve the value of all columns.

Syntax

```
SELECT [{LIMIT offsetnumber | TOP number}] [ALL | DISTINCT] {* | column_expression
[[AS] column_alias] [,column_expression [[AS] column_alias], ...]}
```

where:

`LIMIT offset number`

creates the result set for the Select statement first and then discards the first number of rows specified by *offset* and returns the number of remaining rows specified by *number*. To not discard any of the rows, specify `0` for *offset*, for example, `LIMIT 0` *number*. To discard the first *offset* number of rows and return all the remaining rows, specify `0` for *number*, for example, `LIMIT` *offset*`0`.

`TOP` *number*

is equivalent to `LIMIT 0`*number*.

column_expression

can be simply a column name (for example, `last_name`). More complex expressions may include mathematical operations or string manipulation (for example, `salary * 1.05`). See "SQL expressions" for details. *column_expression* can also include aggregate functions. See "Aggregate functions" for details.

column_alias

can be used to give the column a descriptive name. For example, to assign the alias department to the column dep:

```
SELECT dep AS department FROM emp
```

`DISTINCT`

eliminates duplicate rows from the result of a query. This operator can precede the first column expression. For example:

```
SELECT DISTINCT dep FROM emp
```

Notes

- Separate multiple column expressions with commas (for example, `SELECT last_name, first_name, hire_date`).
- Column names can be prefixed with the table name or table alias. For example, `SELECT emp.last_name` or `e.last_name`, where `e` is the alias for the table `emp`.
- NULL values are not treated as distinct from each other. The default behavior is that all result rows be returned, which can be made explicit with the keyword `ALL`.

See also

SQL expressions on page 104
Aggregate functions on page 94

Aggregate functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of rows. An aggregate can be used with a column name (for example, `AVG(salary)`) or in combination with a more complex column expression (for example, `AVG(salary * 1.07)`).

The following table lists supported aggregate functions.

Note: Doubly nested aggregates, such as `SUM(COUNT(col1))`, are currently not permitted by the driver.

Table 11: Aggregate Functions

Aggregate	Returns
AVG	The average of the values in a numeric column expression. For example, `AVG(salary)` returns the average of all salary column values.
COUNT	The number of values in any field expression. For example, `COUNT(name)` returns the number of name values. When using `COUNT` with a field name, `COUNT` returns the number of non-NULL column values. A special example is `COUNT(*)`, which returns the number of rows in the set, including rows with NULL values. **Note:** The driver does not support `COUNT(DISTINCT *)`. For example, `SELECT COUNT(DISTINCT *) FROM mytable` results in a syntax error.
MAX	The maximum value in any column expression. For example, `MAX(salary)` returns the maximum salary column value.
MIN	The minimum value in any column expression. For example, `MIN(salary)` returns the minimum salary column value.
SUM	The total of the values in a numeric column expression. For example, `SUM(salary)` returns the sum of all salary column values.

Example

The following example uses the `COUNT`, `MAX`, and `AVG` aggregate functions:

```
SELECT
     COUNT(amount) AS numOpportunities,
     MAX(amount) AS maxAmount,
     AVG(amount) AS avgAmount
FROM opportunity o INNER JOIN user u
     ON o.ownerId = u.id
WHERE o.isClosed = 'false' AND
     u.name = 'MyName'
```

From clause

Purpose

The From clause indicates the tables to be used in the Select statement.

Syntax

```
FROM table_name [table_alias] [,...]
```

where:

table_name

is the name of a table or a subquery. Multiple tables define an implicit inner join among those tables. Multiple table names must be separated by a comma. For example:

```
SELECT * FROM emp, dep
```

Subqueries can be used instead of table names. Subqueries must be enclosed in parentheses. See "Subquery in a From clause" for an example.

table_alias

is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias.

Example

The following example specifies two table aliases, e for emp and d for dep:

```
SELECT e.name, d.deptName
FROM emp e, dep d
WHERE e.deptId = d.id
```

table_alias is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias. For example, given the table specification:

```
FROM emp E
```

you may refer to the last_name field as E.last_name. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

See also

Subquery in a From clause on page 96

Join in a From clause

Purpose

You can use a Join as a way to associate multiple tables within a Select statement. Joins may be either explicit or implicit. For example, the following is the example from the previous section restated as an explicit inner join:

```
SELECT * FROM emp INNER JOIN dep ON id=empId
SELECT e.name, d.deptName
FROM emp e INNER JOIN dep d ON e.deptId = d.id;
```

whereas the following is the same statement as an implicit inner join:

```
SELECT * FROM emp, dep WHERE emp.deptID=dep.id
```

Note: The ON clause in a join expression must evaluate to a true or false value.

Syntax

```
FROM table_name {RIGHT OUTER | INNER | LEFT OUTER | CROSS | FULL OUTER} JOIN table.key
 ON search-condition
```

Example

In this example, two tables are joined using LEFT OUTER JOIN. T1, the first table named includes nonmatching rows.

```
SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.key = T2.key
```

If you use a CROSS JOIN, no ON expression is allowed for the join.

See also

Subqueries on page 102

Subquery in a From clause

Subqueries can be used in the From clause in place of table references (*table_name*).

Example

```
SELECT * FROM (SELECT * FROM emp WHERE sal > 10000) new_emp, dept WHERE
new_emp.deptno = dept.deptno
```

See also

SQL expressions on page 104

Where clause

Purpose

Specifies the conditions that rows must meet to be retrieved.

Syntax

WHERE *expr1 rel_operator expr2*

where:

expr1

is either a column name, literal, or expression.

expr2

is either a column name, literal, expression, or subquery. Subqueries must be enclosed in parentheses.

rel_operator

is the relational operator that links the two expressions.

Example

The following Select statement retrieves the first and last names of employees that make at least $20,000.

```
SELECT last_name, first_name FROM emp WHERE salary >= 20000
```

Group By clause

Purpose

Specifies the names of one or more columns by which the returned values are grouped. This clause is used to return a set of aggregate values.

Syntax

```
GROUP BY column_expression [,...]
```

where:

column_expression

is either a column name or a SQL expression. Multiple values must be separated by a comma. If *column_expression* is a column name, it must match one of the column names specified in the Select clause. Also, the Group By clause must include all non-aggregate columns specified in the Select list.

Example

The following example totals the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

See also

Subqueries on page 102
SQL expressions on page 104

Having clause

Purpose

Specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a Group By clause.

Syntax

```
HAVING expr1 rel_operator expr2
```

where:

expr1 | expr2

can be column names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause. See "SQL expressions" for details regarding SQL expressions.

rel_operator

is the relational operator that links the two expressions.

Example

The following example returns only the departments that have salaries totaling more than $200,000:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id HAVING sum(salary) > 200000
```

See also

Subqueries on page 102
SQL expressions on page 104

Union operator

Purpose

Combines the results of two Select statements into a single result. The single result is all the returned rows from both Select statements. By default, duplicate rows are not returned. To return duplicate rows, use the All keyword (`UNION ALL`).

Syntax

```
select_statement
UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT]} | INTERSECT
[DISTINCT]select_statement
```

Notes

- When using the Union operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

Intersect operator

Purpose

Intersect operator returns a single result set. The result set contains rows that are returned by both Select statements. Duplicates are returned unless the Distinct operator is added.

Syntax

```
select_statement
INTERSECT [DISTINCT]
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using the Intersect operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
INTERSECT [DISTINCT]
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
INTERSECT
SELECT salary, last_name FROM raises
```

Except and Minus operators

Purpose

Return the rows from the left Select statement that are not included in the result of the right Select statement.

Syntax

```
select_statement
{EXCEPT [DISTINCT] | MINUS [DISTINCT]}
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using one of these operators, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
EXCEPT
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
EXCEPT
SELECT salary, last_name FROM raises
```

Order By clause

Purpose

The Order By clause specifies how the rows are to be sorted.

Syntax

`ORDER BY sort_expression [DESC | ASC] [,...]`

where:

sort_expression

is either the name of a column, a column alias, a SQL expression, or the positioned number of the column or expression in the select list to use.

The default is to perform an ascending (ASC) sort.

Example

To sort by `last_name` and then by `first_name`, you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, `last_name` is the second item in the Select list, so `ORDER BY 2,3` sorts by `last_name` and then by `first_name`.

See also

Subqueries on page 102
SQL expressions on page 104

Limit clause

Purpose

Places an upper bound on the number of rows returned in the result.

Syntax

`LIMIT` *number_of_rows* `[OFFSET` *offset_number*`]`

where:

number_of_rows

specifies a maximum number of rows in the result. A negative number indicates no upper bound.

`OFFSET`

specifies how many rows to skip at the beginning of the result set. *offset_number* is the number of rows to skip.

Notes

- In a compound query, the Limit clause can appear only on the final Select statement. The limit is applied to the entire query, not to the individual Select statement to which it is attached.

Example

The following example returns a maximum of 20 rows.

```
SELECT last_name, first_name FROM emp WHERE salary > 20000 ORDER BY dept_idc LIMIT
20
```

Subqueries

A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level query is called a Select statement, and a query nested within a Select statement is called a subquery.

A subquery is a query expression that appears in the body of another expression such as a Select, an Update, or a Delete statement. In the following example, the second Select statement is a subquery:

```
SELECT * FROM emp WHERE deptno IN (SELECT deptno FROM dept)
```

IN predicate

Purpose

The In predicate specifies a set of values against which to compare a result set. If the values are being compared against a subquery, only a single column result set is returned.

Syntax

value [NOT] IN (*value1*, *value2*,...)

OR

value [NOT] IN (*subquery*)

Example

```
SELECT * FROM emp WHERE deptno IN
(SELECT deptno FROM dept WHERE dname <> 'Sales')
```

EXISTS predicate

Purpose

The Exists predicate is true only if the cardinality of the subquery is greater than 0; otherwise, it is false.

Syntax

EXISTS (*subquery*)

Example

```
SELECT empno, ename, deptno FROM emp e WHERE EXISTS
(SELECT deptno FROM dept WHERE e.deptno = dept.deptno)
```

UNIQUE predicate

Purpose

The Unique predicate is used to determine whether duplicate rows exist in a virtual table (one returned from a subquery).

Syntax

```
UNIQUE (subquery)
```

Example

```
SELECT * FROM dept d WHERE UNIQUE
(SELECT deptno FROM emp e WHERE e.deptno = d.deptno)
```

Correlated subqueries

Purpose

A correlated subquery is a subquery that references a column from a table referred to in the parent statement. A correlated subquery is evaluated once for each row processed by the parent Select statement.

A correlated subquery answers a multiple-part question in which the answer depends on the value in each row processed by the parent statement. For example, you can use a correlated subquery to determine which employees earn more than the average salaries for their departments. In this case, the correlated subquery specifically computes the average salary for each department.

Syntax

```
SELECT select_list
   FROM table1 t_alias1
   WHERE expr rel_operator
   (SELECT column_list
       FROM table2 t_alias2
       WHERE t_alias1.columnrel_operatort_alias2.column)
```

Notes

- Correlated column names in correlated subqueries must be explicitly qualified with the table name of the parent.

Example A

The following statement returns data about employees whose salaries exceed their department average. This statement assigns an alias to emp, the table containing the salary information, and then uses the alias in a correlated subquery:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
  ORDER BY deptno
```

Example B

This is an example of a correlated subquery that returns row values:

```
SELECT * FROM dept "outer" WHERE 'manager' IN
  (SELECT managername FROM emp
  WHERE "outer".deptno = emp.deptno)
```

Example C

This is an example of finding the department number (`deptno`) with multiple employees:

```
SELECT * FROM dept main WHERE 1 <
  (SELECT COUNT(*) FROM emp WHERE deptno = main.deptno)
```

Example D

This is an example of correlating a table with itself:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
```

SQL expressions

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. You can use expressions in the Where, and Having of Select statements; and in the Set clauses of Update statements.

Expressions enable you to use mathematical operations as well as character string manipulation operators to form complex queries.

The driver supports both unquoted and quoted identifiers. An unquoted identifier must start with an ASCII alpha character and can be followed by zero

Quoted identifiers must be enclosed in double quotation marks (""). A quoted identifier can contain any Unicode character including the space character. The driver recognizes the Unicode escape sequence \uxxxx as a Unicode character. You can specify a double quotation mark in a quoted identifier by escaping it with a double quotation mark.

The maximum length of both quoted and unquoted identifiers is 128 characters.

Valid expression elements are:

- Column names
- Literals
- Operators
- Functions

Column names

The most common expression is a simple column name. You can combine a column name with other expression elements.

Literals

Literals are fixed data values. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant. Literals are classified into types, including the following:

- Binary
- Character string
- Date
- Floating point
- Integer
- Numeric
- Time
- Timestamp

The following table describes the literal format for supported SQL data types.

Table 12: Literal Syntax Examples

SQL Type	Literal Syntax	Example
BIGINT	*n* where *n* is any valid integer value in the range of the INTEGER data type	`12 or -34 or 0`
BOOLEAN	Min Value: 0 Max Value: 1	`0` `1`
DATE	DATE'*date*'	`'2010-05-21'`
DATETIME	TIMESTAMP'*ts*'	`'2010-05-21 18:33:05.025'`

DECIMAL	*n.f*
	where:
	n
	is the integral part
	f
	is the fractional part
		`0.25`
		`3.1415`
		`-7.48`

DOUBLE	*n.f*E*x* where: *n* is the integral part *f* is the fractional part *x* is the exponent	`1.2E0` or `2.5E40` or `-3.45E2` or `5.67E-4`

SQL Type	Literal Syntax	Example
INTEGER	*n* where *n* is a valid integer value in the range of the INTEGER data type	12 or -34 or 0
LONGVARBINARY	'*hex_value*'	'000482ff'
LONGVARCHAR	'*value*'	'This is a string literal'
TIME	TIME'*time*'	'2010-05-21 18:33:05.025'
VARCHAR	'*value*'	'This is a string literal'

Character string literals

Text specifies a character string literal. A character string literal must be enclosed in single quotation marks. To represent one single quotation mark within a literal, you must enter two single quotation marks. When the data in the fields is returned to the client, trailing blanks are stripped.

A character string literal can have a maximum length of 32 KB, that is, (32*1024) bytes.

Example

```
'Hello'
'Jim''s friend is Joe'
```

Numeric literals

Unquoted numeric values are treated as numeric literals. If the unquoted numeric value contains a decimal point or exponent, it is treated as a real literal; otherwise, it is treated as an integer literal.

Example

```
+1894.1204
```

Binary literals

Binary literals are represented with single quotation marks. The valid characters in a binary literal are 0-9, a-f, and A-F.

Example

```
'00af123d'
```

Date/Time literals

Date and time literal values are enclosed in single quotion marks ('*value*').

- The format for a Date literal is DATE'*date*'.

- The format for a Time literal is TIME'*time*'.
- The format for a Timestamp literal is TIMESTAMP'*ts*'.

Integer literals

Integer literals are represented by a string of numbers that are not enclosed in quotation marks and do not contain decimal points.

Notes

- Integer constants must be whole numbers; they cannot contain decimals.
- Integer literals can start with sign characters (+/-).

Example

```
1994 or -2
```

Operators

This section describes the operators that can be used in SQL expressions.

Note: Numeric operators are restricted to numeric types. Numeric operators do not support non-numeric types.

Unary operator

A unary operator operates on only one operand.

```
operator operand
```

Binary operator

A binary operator operates on two operands.

```
operand1 operator operand2
```

If an operator is given a null operand, the result is always null. The only operator that does not follow this rule is concatenation (||).

Arithmetic operators

You can use an arithmetic operator in an expression to negate, add, subtract, multiply, and divide numeric values. The result of this operation is also a numeric value. The + and - operators are also supported in date/time fields to allow date arithmetic. The following table lists the supported arithmetic operators.

Table 13: Arithmetic Operators

Operator	Purpose	Example
+ -	Denotes a positive or negative expression. These are unary operators.	`SELECT * FROM emp WHERE comm = -1`

Operator	Purpose	Example
* /	Multiplies, divides. These are binary operators.	`UPDATE emp SET sal = sal + sal * 0.10`
+ -	Adds, subtracts. These are binary operators.	`SELECT sal + comm FROM emp WHERE empno > 100`

Concatenation operator

The concatenation operator manipulates character strings. The following table lists the only supported concatenation operator.

Table 14: Concatenation Operator

Operator	Purpose	Example
||	Concatenates character strings.	`SELECT 'Name is' || ename FROM emp`

The result of concatenating two character strings is the data type VARCHAR.

Comparison operators

Comparison operators compare one expression to another. The result of such a comparison can be TRUE, FALSE, or UNKNOWN (if one of the operands is NULL). The driver considers the UNKNOWN result as FALSE.

The following table lists the supported comparison operators.

Table 15: Comparison Operators

Operator	Purpose	Example
=	Equality test.	`SELECT * FROM emp WHERE sal = 1500`
!=<>	Inequality test.	`SELECT * FROM emp WHERE sal != 1500`
><	"Greater than" and "less than" tests.	`SELECT * FROM emp WHERE sal > 1500 SELECT * FROM emp WHERE sal < 1500`
>=<=	"Greater than or equal to" and "less than or equal to" tests.	`SELECT * FROM emp WHERE sal >= 1500 SELECT * FROM emp WHERE sal <= 1500`

Operator	Purpose	Example
LIKE	% and _ wildcards can be used to search for a pattern in a column. The percent sign denotes zero, one, or multiple characters, while the underscore denotes a single character. The right-hand side of a LIKE expression must evaluate to a string or binary.	`SELECT * FROM emp WHERE ENAME LIKE 'J%'`

ESCAPE clause in LIKE operator
LIKE 'pattern string' ESCAPE 'c'
	The Escape clause is supported in the LIKE predicate to indicate the escape character. Escape characters are used in the pattern string to indicate that any wildcard character that is after the escape character in the pattern string should be treated as a regular character.
	The default escape character is backslash (\).
		`SELECT * FROM emp WHERE ENAME LIKE 'J%\_%' ESCAPE '\'`
		This matches all records with names that start with letter 'J' and have the '_' character in them.
		`SELECT * FROM emp WHERE ENAME LIKE 'JOE\_JOHN' ESCAPE '\'`
		This matches only records with name 'JOE_JOHN'.

[NOT] IN	"Equal to any member of" test.	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)`
[NOT] BETWEEN x AND y	"Greater than or equal to x" and "less than or equal to y."	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000`
EXISTS	Tests for existence of rows in a subquery.	`SELECT empno, ename, deptno FROM emp e WHERE EXISTS (SELECT deptno FROM dept WHERE e.deptno = dept.deptno)`
IS [NOT] NULL	Tests whether the value of the column or expression is NULL.	`SELECT * FROM emp WHERE ename IS NOT NULL SELECT * FROM emp WHERE ename IS NULL`

Logical operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 16: Logical Operators

Operator	Purpose	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	`SELECT * FROM emp WHERE NOT (job IS NULL)` `SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000)`
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10`
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10`

Example

In the Where clause of the following Select statement, the AND logical operator is used to ensure that managers earning more than $1000 a month are returned in the result:

```
SELECT * FROM emp WHERE jobtitle = manager AND sal > 1000
```

Operator precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. The following table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.You can change the order of precedence by using parentheses. Enclosing expressions in parentheses forces them to be evaluated together.

Table 17: Operator Precedence

Precedence	Operator
1	+ (Positive), - (Negative)
2	*(Multiply), / (Division)
3	+ (Add), - (Subtract)
4	|| (Concatenate)
5	=, >, <, >=, <=, <>, != (Comparison operators)
6	NOT, IN, LIKE
7	AND
8	OR

Example A

The query in the following example returns employee records for which the department number is 1 or 2 and the salary is greater than $1000:

```
SELECT * FROM emp WHERE (deptno = 1 OR deptno = 2) AND sal > 1000
```

Because parenthetical expressions are forced to be evaluated first, the OR operation takes precedence over AND.

Example B

In the following example, the query returns records for all the employees in department 1, but only employees whose salary is greater than $1000 in department 2.

```
SELECT * FROM emp WHERE deptno = 1 OR deptno = 2 AND sal > 1000
```

The AND operator takes precedence over OR, so that the search condition in the example is equivalent to the expression `deptno = 1 OR (deptno = 2 AND sal > 1000)`.

Functions

The driver supports a number of functions that you can use in expressions, as listed and described in "Scalar functions."

Refer to "Scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

Conditions

A condition specifies a combination of one or more expressions and logical operators that evaluates to either TRUE, FALSE, or UNKNOWN. You can use a condition in the Where clause of the Delete, Select, and Update statements; and in the Having clauses of Select statements. The following describes supported conditions.

Table 18: Conditions

Condition	Description
Simple comparison	Specifies a comparison with expressions or subquery results. `= , != <>, < , >, <=, <=`
Group comparison	Specifies a comparison with any or all members in a list or subquery. `[= , != <>, < , >, <=, <=] [ANY, ALL, SOME]`
Membership	Tests for membership in a list or subquery. `[NOT] IN`
Range	Tests for inclusion in a range. `[NOT] BETWEEN`

Condition	Description
NULL	Tests for nulls. `IS NULL, IS NOT NULL`
EXISTS	Tests for existence of rows in a subquery. `[NOT] EXISTS`
LIKE	Specifies a test involving pattern matching. `[NOT] LIKE`
Compound	Specifies a combination of other conditions. `CONDITION [AND/OR] CONDITION`

7

Introduction to the Google Analytics 4 data model

The driver exposes the Google Analytics data model in the form of a relational schema. Google Analytics stores two types of data: admin data and analytics data. Admin data consists of objects related to the management of Google Analytics services. The driver maps these objects directly as relational tables. Analytics data consists of all the data that Google Analytics collects about a website.

Given the large volume of analytics data available and the limitations around which aspects of analytic data that can be returned in a single result set, it can be challenging to formulate valid SQL queries. To address this issue, the driver requires that you create custom tables containing the specific metrics and dimensions that you want to run queries against. The Configuration Manager includes a table designer tool to assist with custom table creation. See Adding custom tables for details.

Note: The driver itself defines a number of functional tables, such as `INCOMPATIBILITY` and `METADATA`. The driver uses these tables to translate the Google Analytics data model into a relational schema. The metadata for these tables, as well as the admin and custom tables, is provided in corresponding topics beginning with the ACCOUNTS topic.

The following sections provide information to help you write effective queries against Google Analytics.

- Google Analytics data structure
- Adding custom tables
- Paging support
- Realtime reports

Google Analytics data structure

Google Analytics is a service that generates detailed statistics about a website's traffic and traffic sources. But Google Analytics is not just a database. It is a multi-dimensional hypercube containing all kinds of measurements about traffic to a website. When you connect to Google Analytics using the driver, you can reach into this repository and flatten it into relational data that can be used with your JDBC application.

Imagine a very small store of data about your website. For each hit, the Google Analytics service logs the date, language of user, country of origin, new or returning user, and their time on the site (in seconds).

2019-01-01	en	US	new	5.3
2019-01-01	nl	DK	new	90.4
2019-01-01	es	ES	new	24
2019-01-01	ja	JP	new	4.2
2019-01-01	es	MX	new	345.3
2019-01-01	ja	JP	returning	655.9
2019-01-02	en	US	new	45.7
2019-01-02	en	US	new	345.9
2019-01-02	es	ES	new	57.7
2019-01-02	en	US	new	6.8
2019-01-03	es	MX	new	876.1
2019-01-03	ja	JP	returning	5.7
2019-01-03	en	GB	new	5.6
2019-01-03	en	US	new	617.9
2019-01-03	en	US	returning	56.1
2019-01-04	es	MX	new	45.1
2019-01-04	jp	JP	new	178
2019-01-04	en	US	returning	103.9

Google Analytics collected data for this website over four days. The data is broken down by date, language, country and user type. For each visit, the time spent on the site was recorded. The time on the site is a *metric*, and the other columns are *dimensions*.

In an actual scenario, Google Analytics aggregates a vast amount of information about your website. It measures hundreds of things, and categorizes them by hundreds of dimensions. The query interface that Google Analytics provides allows you to fetch these metrics and group them. Because of the massive amount of information they store, their interface limits you to fetching up to ten metrics at a time, grouped by no more than nine dimensions.

For example, suppose you want to know how much time new visitors spent on the site. Your dimension is user type and your metric is time. You would get back two rows:

new	2648
returning	821.6

How much data you get back depends on how you ask for it. If you ask for two dimensions, you get even more data, because you get one row per permutation. Requesting how much time users have spent on each day, broken down by country, returns more rows:

2019-01-01	DK	90.4
2019-01-01	ES	24
2019-01-01	JP	660.1
2019-01-01	MX	345.3
2019-01-01	US	5.3
2019-01-02	ES	57.7
2019-01-02	US	398.4
2019-01-03	GB	5.6
2019-01-03	JP	5.7
2019-01-03	MX	876.1
2019-01-03	US	674
2019-01-04	JP	178
2019-01-04	MX	45.1
2019-01-04	US	103.9

Adding custom tables

Custom tables can be added to your Google Analytics relational schema, and further customized, using the Configuration Manager. Once you open the Configuration Manager, you can add and customize tables from the **Configure Logical Schema** button on the **Schema Settings** tab. This tool allows you define a new table, or customize an existing table, in terms of metrics and dimensions. You can select up to ten metrics and nine dimensions per table. Based on your selections, a table definition is created in the form of a JSON string which is incorporated into the connection string as a value for the Add Tables (AddTables) option. For example:

```
AddTables='{"MyTable":["sessions","_language","_country"]}'
```

Note: Dimension names specified in the string must include a leading underscore character. For example, `"_browser"`. The driver uses the underscore character to distinguish dimensions from metrics.

The table that results from this example would include the three selected columns, plus the columns `currencyCode`, `keepEmptyRows`, `propertyId`, `rowId`, and `timeZone`. The following is an example query for your custom table:

```
SELECT _LANGUAGE,SESSIONS FROM MyTable
```

Paging support

The driver employs row offset paging when returning results using the Reports API. The page size returned for results is set to 10,000 rows.

Realtime reports

Realtime reports return events and usage data for periods ranging from the present to 30 minutes ago (up to 60 minutes for Google Analytics 360 properties). Reports can return up to 14 dimensions and 4 metrics. For more information on the supported dimensions and metrics, refer to the documentation for the Google Analytics Data API.

You can configure Realtime reports by specifying the following parameters for the Default Query Options (DefaultQueryOptions) option.

- `startMinutesAgo`: The inclusive start minutes for the query with Realtime Report API. The default is `29` (twenty-nine minutes prior to the current time).
- `endMinutesAgo`: The inclusive end minutes for the query with Realtime report API. The default is `0` (current time).

These filters will automatically be applied to your query. For example, the following query can be used to return realtime reports:

```
SELECT _appVersion,_city,activeUsers,eventCount FROM REALTIMEREPORTS where
propertyID=123344545
```

Note that paging is not supported when querying realtime reports, and a maximum of 100,000 rows can be returned.

For details, see the following topics:

- ACCOUNTS
- ACCOUNTS_DATASHARINGSETTINGS
- ACCOUNTS_SEARCHCHANGEHISTORYEVENTS
- ACCOUNTS_SEARCHCHANGEHISTORYEVENTS_CHANGES
- ACCOUNTS_USERLINKS
- ACCOUNTS_USERLINKS_AUDIT
- ACCOUNTS_USERLINKS_AUDIT_DIRECTROLES
- ACCOUNTS_USERLINKS_AUDIT_EFFECTIVEROLES
- ACCOUNTS_USERLINKS_DIRECTROLES
- ACCOUNTS_USERLINKS_NAMES
- ACCOUNTSLIST
- ACCOUNTSUMMARIES

- ACCOUNTSUMMARIES_PROPERTYSUMMARIES
- INCOMPATIBILITY
- METADATA
- METADATA_DIMENSIONS
- METADATA_DIMENSIONS_DEPRECATEDAPINAMES
- METADATA_METRICS
- METADATA_METRICS_BLOCKEDREASONS
- METADATA_METRICS_DEPRECATEDAPINAMES
- PROPERTIES
- PROPERTIES_ACCESSREPORT
- PROPERTIES_ACCESSREPORT_DIMENSIONHEADERS
- PROPERTIES_ACCESSREPORT_METRICHEADERS
- PROPERTIES_ACCESSREPORT_ROWS
- PROPERTIES_ACKNOWLEDGEUSERDATACOLLECTION
- PROPERTIES_ATTRIBUTIONSETTINGS
- PROPERTIES_AUDIENCES
- PROPERTIES_AUDIENCES_FILTERCLAUSES
- PROPERTIES_AUDIENCES_SEQUENCESTEPS
- PROPERTIES_CONVERSIONEVENTS
- PROPERTIES_CUSTOMDIMENSIONS
- PROPERTIES_CUSTOMMETRICS
- PROPERTIES_CUSTOMMETRICS_RESTRICTEDMETRICTYPE
- PROPERTIES_DATARETENTIONSETTINGS
- PROPERTIES_DATASTREAMS
- PROPERTIES_DATASTREAMS_GLOBALSITETAG
- PROPERTIES_DATASTREAMS_MEASUREMENTPROTOCOLSECRETS
- PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKS
- PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKPROPOSALS
- PROPERTIES_FIREBASELINKS
- PROPERTIES_GOOGLEADSLINKS
- PROPERTIES_GOOGLESIGNALSSETTINGS
- PROPERTIES_USERLINKS
- PROPERTIES_USERLINKS_AUDIT

- PROPERTIES_USERLINKS_AUDIT_DIRECTROLES
- PROPERTIES_USERLINKS_AUDIT_EFFECTIVEROLES
- PROPERTIES_USERLINKS_DIRECTROLES
- PROPERTIESLIST
- REALTIMEREPORTS

ACCOUNTS

Endpoint

`https://analyticsadmin.googleapis.com/v1/accounts/{accountId}`

Columns

The ACCOUNTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Note: When querying the ACCOUNTS table, you must filter by the ACCOUNTID to return results; otherwise, the service returns a `400 Bad Request` error. For example:

```
SELECT * FROM ACCOUNTS WHERE ACCOUNTID = '12345679' LIMIT 10
```

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNTID	VarChar(64)
CREATETIME	Timestamp(6)
DISPLAYNAME	VarChar(64)
REGIONCODE	VarChar(64)
UPDATETIME	Timestamp(6)

ACCOUNTS_DATASHARINGSETTINGS

Endpoint

`https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/dataSharingSettings`

Columns

The ACCOUNTS_DATASHARINGSETTINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNTID	VarChar(64)
SHARINGWITHGOOGLEANYSALESENABLED	Boolean
SHARINGWITHGOOGLEASSIGNEDSALESENABLED	Boolean
SHARINGWITHGOOGLEPRODUCTSENABLED	Boolean
SHARINGWITHGOOGLESUPPORTENABLED	Boolean
SHARINGWITHOTHERSENABLED	Boolean

ACCOUNTS_SEARCHCHANGEHISTORYEVENTS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}:searchChangeHistoryEvents
```

Columns

The ACCOUNTS_SEARCHCHANGEHISTORYEVENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	VarChar(64)
ACCOUNTID	VarChar(64)
ACTORTYPE	VarChar(22)
CHANGESFILTERED	Boolean
CHANGETIME	VarChar(64)
USERACTOREMAIL	VarChar(64)

ACCOUNTS_SEARCHCHANGEHISTORYEVENTS_CHANGES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}:searchChangeHistoryEvents
```

Parent Table

ACCOUNTS_SEARCHCHANGEHISTORYEVENTS

Columns

The ACCOUNTS_SEARCHCHANGEHISTORYEVENTS_CHANGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTS_SEARCHCHANGEHISTORYEVENTS_ID*	VarChar(64)	References: ACCOUNTS_SEARCHCHANGEHISTORYEVENTS.ID
POSITION*	Integer	
ACCOUNTID	VarChar(64)	
ACTION	VarChar(23)	
RESOURCE	VarChar(64)	
RESOURCEAFTERCHANGE_ACCOUNT_CREATETIME	Timestamp(6)	
RESOURCEAFTERCHANGE_ACCOUNT_DELETED	Boolean	
RESOURCEAFTERCHANGE_ACCOUNT_DISPLAYNAME	VarChar(64)	
RESOURCEAFTERCHANGE_ACCOUNT_NAME	VarChar(64)	
RESOURCEAFTERCHANGE_ACCOUNT_REGIONCODE	VarChar(64)	
RESOURCEAFTERCHANGE_ACCOUNT_UPDATETIME	Timestamp(6)	
RESOURCEAFTERCHANGE_CONVERSIONEVENT_CREATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_CONVERSIONEVENT_CUSTOM	Boolean	

Column Name	Data Type	Notes
RESOURCEAFTERCHANGE _CONVERSIONEVENT_DELETABLE	Boolean	
RESOURCEAFTERCHANGE _CONVERSIONEVENT_EVENTNAME	VarChar(64)	
RESOURCEAFTERCHANGE _CONVERSIONEVENT_NAME	VarChar(64)	
RESOURCEAFTERCHANGE _DATARETENTIONSETTINGS _EVENTDATARETENTION	VarChar(30)	
RESOURCEAFTERCHANGE _DATARETENTIONSETTINGS_NAME	VarChar(64)	
RESOURCEAFTERCHANGE _DATARETENTIONSETTINGS _RESETUSERDATAONNEWACTIVITY	Boolean	
RESOURCEAFTERCHANGE_DATASTREAM _ANDROIDAPPSTREAMDATA _FIREBASEAPPID	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _ANDROIDAPPSTREAMDATA _PACKAGENAME	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _CREATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _DISPLAYNAME	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _IOSAPPSTREAMDATA_BUNDLEID	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _IOSAPPSTREAMDATA_FIREBASEAPPID	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _NAME	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _TYPE	VarChar(30)	
RESOURCEAFTERCHANGE_DATASTREAM _UPDATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM _WEBSTREAMDATA_DEFAULTURI	VarChar(64)	

Column Name	Data Type	Notes
RESOURCEAFTERCHANGE_DATASTREAM_WEBSTREAMDATA_FIREBASEAPPID	VarChar(64)	
RESOURCEAFTERCHANGE_DATASTREAM_WEBSTREAMDATA_MEASUREMENTID	VarChar(64)	
RESOURCEAFTERCHANGE_FIREBASELINK_CREATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_FIREBASELINK_NAME	VarChar(64)	
RESOURCEAFTERCHANGE_FIREBASELINK_PROJECT	VarChar(64)	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_ADSPERSONALIZATIONENABLED	Boolean	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_CANMANAGECLIENTS	Boolean	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_CREATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_CREATOREMAILADDRESS	VarChar(64)	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_CUSTOMERID	VarChar(64)	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_NAME	VarChar(64)	
RESOURCEAFTERCHANGE_GOOGLEADSLINK_UPDATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_MEASUREMENTPROTOCOLSECRET_DISPLAYNAME	VarChar(64)	
RESOURCEAFTERCHANGE_MEASUREMENTPROTOCOLSECRET_NAME	VarChar(64)	
RESOURCEAFTERCHANGE_MEASUREMENTPROTOCOLSECRET_SECRETVALUE	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_ACCOUNT	VarChar(64)	

Column Name	Data Type	Notes
RESOURCEAFTERCHANGE_PROPERTY_CREATETIME	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_CURRENCYCODE	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_DELETETIME	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_DISPLAYNAME	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_EXPIRETIME	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_INDUSTRYCATEGORY	VarChar(30)	
RESOURCEAFTERCHANGE_PROPERTY_NAME	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_PARENT	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_PROPERTYTYPE	VarChar(30)	
RESOURCEAFTERCHANGE_PROPERTY_SERVICELEVEL	VarChar(30)	
RESOURCEAFTERCHANGE_PROPERTY_TIMEZONE	VarChar(64)	
RESOURCEAFTERCHANGE_PROPERTY_UPDATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_ACCOUNT_CREATETIME	Timestamp(6)	
RESOURCEBEFORECHANGE_ACCOUNT_DELETED	Boolean	
RESOURCEBEFORECHANGE_ACCOUNT_DISPLAYNAME	VarChar(64)	
RESOURCEBEFORECHANGE_ACCOUNT_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_ACCOUNT_REGIONCODE	VarChar(64)	

Column Name	Data Type	Notes
RESOURCEBEFORECHANGE_ACCOUNT_UPDATETIME	Timestamp(6)	
RESOURCEBEFORECHANGE_CONVERSIONEVENT_CREATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_CONVERSIONEVENT_CUSTOM	Boolean	
RESOURCEBEFORECHANGE_CONVERSIONEVENT_DELETABLE	Boolean	
RESOURCEBEFORECHANGE_CONVERSIONEVENT_EVENTNAME	VarChar(64)	
RESOURCEBEFORECHANGE_CONVERSIONEVENT_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_DATARETENTIONSETTINGS_EVENTDATARETENTION	VarChar(30)	
RESOURCEBEFORECHANGE_DATARETENTIONSETTINGS_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_DATARETENTIONSETTINGS_RESETUSERDATAONNEWACTIVITY	Boolean	
RESOURCEBEFORECHANGE_DATASTREAM_ANDROIDAPPSTREAMDATA_FIREBASEAPPID	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_ANDROIDAPPSTREAMDATA_PACKAGENAME	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_CREATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_DISPLAYNAME	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_IOSAPPSTREAMDATA_BUNDLEID	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_IOSAPPSTREAMDATA_FIREBASEAPPID	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_NAME	VarChar(64)	

Column Name	Data Type	Notes
RESOURCEBEFORECHANGE_DATASTREAM_TYPE	VarChar(30)	
RESOURCEBEFORECHANGE_DATASTREAM_UPDATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_WEBSTREAMDATA_DEFAULTURI	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_WEBSTREAMDATA_FIREBASEAPPID	VarChar(64)	
RESOURCEBEFORECHANGE_DATASTREAM_WEBSTREAMDATA_MEASUREMENTID	VarChar(64)	
RESOURCEBEFORECHANGE_FIREBASELINK_CREATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_FIREBASELINK_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_FIREBASELINK_PROJECT	VarChar(64)	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_ADSPERSONALIZATIONENABLED	Boolean	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_CANMANAGECLIENTS	Boolean	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_CREATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_CREATOREMAILADDRESS	VarChar(64)	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_CUSTOMERID	VarChar(64)	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_GOOGLEADSLINK_UPDATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_MEASUREMENTPROTOCOLSECRET_DISPLAYNAME	VarChar(64)	

Column Name	Data Type	Notes
RESOURCEBEFORECHANGE_MEASUREMENTPROTOCOLSECRET_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_MEASUREMENTPROTOCOLSECRET_SECRETVALUE	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_ACCOUNT	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_CREATETIME	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_CURRENCYCODE	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_DELETETIME	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_DISPLAYNAME	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_EXPIRETIME	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_INDUSTRYCATEGORY	VarChar(30)	
RESOURCEBEFORECHANGE_PROPERTY_NAME	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_PARENT	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_PROPERTYTYPE	VarChar(30)	
RESOURCEBEFORECHANGE_PROPERTY_SERVICELEVEL	VarChar(30)	
RESOURCEBEFORECHANGE_PROPERTY_TIMEZONE	VarChar(64)	
RESOURCEBEFORECHANGE_PROPERTY_UPDATETIME	VarChar(64)	

ACCOUNTS_USERLINKS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:batchGet
```

Columns

The ACCOUNTS_USERLINKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNTID	VarChar(64)
EMAILADDRESS	VarChar(64)

ACCOUNTS_USERLINKS_AUDIT

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:audit
```

Columns

The ACCOUNTS_USERLINKS_AUDIT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNTID	VarChar(64)
EMAILADDRESS	VarChar(64)

ACCOUNTS_USERLINKS_AUDIT_DIRECTROLES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:audit
```

Parent Table

ACCOUNTS_USERLINKS_AUDIT

Columns

The ACCOUNTS_USERLINKS_AUDIT_DIRECTROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTS_USERLINKS_AUDIT_NAME*	VarChar(64)	References: ACCOUNTS_USERLINKS_AUDIT .NAME
POSITION*	Integer	
ACCOUNTID	VarChar(64)	
ACCOUNTS_USERLINKS_AUDIT _DIRECTROLE	VarChar(64)	

ACCOUNTS_USERLINKS_AUDIT_EFFECTIVEROLES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:audit
```

Parent Table

ACCOUNTS_USERLINKS_AUDIT

Columns

The ACCOUNTS_USERLINKS_AUDIT_EFFECTIVEROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTS_USERLINKS_AUDIT_NAME*	VarChar(64)	References: ACCOUNTS_USERLINKS_AUDIT .NAME
POSITION*	Integer	
ACCOUNTID	VarChar(64)	
ACCOUNTS_USERLINKS_AUDIT _EFFECTIVEROLE	VarChar(64)	

ACCOUNTS_USERLINKS_DIRECTROLES

Endpoint

`https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:batchGet`

Parent Table

ACCOUNTS_USERLINKS

Columns

The ACCOUNTS_USERLINKS_DIRECTROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTS_USERLINKS_NAME*	VarChar(64)	References: ACCOUNTS_USERLINKS.NAME
POSITION*	Integer	
ACCOUNTID	VarChar(64)	
ACCOUNTS_USERLINKS_DIRECTROLE	VarChar(64)	

ACCOUNTS_USERLINKS_NAMES

Endpoint

`https://analyticsadmin.googleapis.com/v1/accounts/{accountId}/userLinks:batchGet`

Parent Table

ACCOUNTS_USERLINKS

Columns

The ACCOUNTS_USERLINKS_NAMES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTS_USERLINKS_NAME*	VarChar(64)	References: ACCOUNTS_USERLINKS.NAME
POSITION*	Integer	
ACCOUNTID	VarChar(64)	
ACCOUNTS_USERLINKS_NAME_1	VarChar(64)	

ACCOUNTSLIST

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accounts
```

Columns

The ACCOUNTSLIST table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNTID	VarChar(64)
CREATETIME	Timestamp(6)
DISPLAYNAME	VarChar(64)
REGIONCODE	VarChar(64)
SHOWDELETED	Boolean
UPDATETIME	Timestamp(6)

ACCOUNTSUMMARIES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/accountSummaries
```

Columns

The ACCOUNTSUMMARIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNT	VarChar(64)
DISPLAYNAME	VarChar(64)

ACCOUNTSUMMARIES_PROPERTYSUMMARIES

Endpoint

`https://analyticsadmin.googleapis.com/v1/accountSummaries`

Parent Table

ACCOUNTSUMMARIES

Columns

The ACCOUNTSUMMARIES_PROPERTYSUMMARIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ACCOUNTSUMMARIES_NAME*	VarChar(64)	References: ACCOUNTSUMMARIES.NAME
POSITION*	Integer	
DISPLAYNAME	VarChar(64)	
PARENT	VarChar(64)	
PROPERTY	VarChar(64)	
PROPERTYTYPE	VarChar(64)	

INCOMPATIBILITY

Endpoint

`<hostname>/v1/properties/{propertyId}:checkCompatibility`

Columns

The INCOMPATIBILITY table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ROWID*	VarChar(32)
COMPATIBILITYFILTER	VARCHAR(0)
DIMENSIONCOMPATIBILITIES	JSON(16777215)
METRICCOMPATIBILITIES	JSON(16777215)

Column Name	Data Type
PROPERTYID	VarChar(64)
SELECTEDFIELDS	VarChar(32767)

METADATA

Endpoint

<*hostname*>/v1/properties/{propertyId}/metadata

Columns

The METADATA table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
PROPERTYID	VarChar(64)

METADATA_DIMENSIONS

Endpoint

<*hostname*>/v1/properties/{propertyId}/metadata

Parent Table

METADATA

Columns

The METADATA_DIMENSIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
METADATA_NAME*	VarChar(64)	References: METADATA.NAME
POSITION*	Integer	
APINAME	VarChar(64)	
CATEGORY	VarChar(64)	

Column Name	Data Type	Notes
CUSTOMDEFINITION	Boolean	
DESCRIPTION	VarChar(64)	
PROPERTYID	VarChar(64)	
UINAME	VarChar(64)	

METADATA_DIMENSIONS_DEPRECATEDAPINAMES

Endpoint

<*hostname*>/v1/properties/{propertyId}/metadata

Parent Table

METADATA_DIMENSIONS

Columns

The METADATA_DIMENSIONS_DEPRECATEDAPINAMES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
METADATA_NAME*	VarChar(64)	References: METADATA.NAME
METADATA_DIMENSIONS_POSITION*	Integer	References: METADATA_DIMENSIONS.POSITION
POSITION*	Integer	
METADATA_DIMENSIONS _DEPRECATEDAPINAME	VarChar(64)	
PROPERTYID	VarChar(64)	

METADATA_METRICS

Endpoint

<*hostname*>/v1/properties/{propertyId}/metadata

Parent Table

METADATA

Columns

The METADATA_METRICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
METADATA_NAME*	VarChar(64)	References: METADATA.NAME
POSITION*	Integer	
APINAME	VarChar(64)	
CATEGORY	VarChar(64)	
CUSTOMDEFINITION	Boolean	
DESCRIPTION	VarChar(64)	
EXPRESSION	VarChar(64)	
PROPERTYID	VarChar(64)	
TYPE	VarChar(30)	
UINAME	VarChar(64)	

METADATA_METRICS_BLOCKEDREASONS

Endpoint

`<hostname>/v1/properties/{propertyId}/metadata`

Parent Table

METADATA_METRICS

Columns

The METADATA_METRICS_BLOCKEDREASONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
METADATA_NAME*	VarChar(64)	References: METADATA.NAME
METADATA_METRICS_POSITION*	Integer	References: METADATA_METRICS.POSITION
POSITION*	Integer	

Column Name	Data Type	Notes
METADATA_METRICS_BLOCKEDREASON	VarChar(30)	
PROPERTYID	VarChar(64)	

METADATA_METRICS_DEPRECATEDAPINAMES

Endpoint

<*hostname*>/v1/properties/{propertyId}/metadata

Parent Table

METADATA_METRICS

Columns

The METADATA_METRICS_DEPRECATEDAPINAMES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
METADATA_NAME*	VarChar(64)	References: METADATA.NAME
METADATA_METRICS_POSITION*	Integer	References: METADATA_METRICS.POSITION
POSITION*	Integer	
METADATA_METRICS _DEPRECATEDAPINAME	VarChar(64)	
PROPERTYID	VarChar(64)	

PROPERTIES

Endpoint

https://analyticsadmin.googleapis.com/v1/properties/{propertyId}

Columns

The PROPERTIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Note: When querying the PROPERTIES table, you must filter by the PROPERTYID to return results; otherwise, the service returns a `400 Bad Request` error. For example:

```
SELECT * FROM PROPERTIES WHERE PROPERTYID = '12345679' LIMIT 10
```

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNT	VarChar(64)
CREATETIME	VarChar(64)
CURRENCYCODE	VarChar(64)
DELETETIME	VarChar(64)
DISPLAYNAME	VarChar(64)
EXPIRETIME	VarChar(64)
INDUSTRYCATEGORY	VarChar(35)
PARENT	VarChar(64)
PROPERTYID	VarChar(64)
PROPERTYTYPE	VarChar(23)
SERVICELEVEL	VarChar(25)
TIMEZONE	VarChar(64)
UPDATETIME	VarChar(64)

PROPERTIES_ACCESSREPORT

Endpoint

`https://analyticsadmin.googleapis.com/v1/properties/{propertyId}:runAccessReport`

Columns

The PROPERTIES_ACCESSREPORT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
QUOTA	VarChar(255)
ROWCOUNT	Integer

PROPERTIES_ACCESSREPORT_DIMENSIONHEADERS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}:runAccessReport
```

Parent Table

PROPERTIES_ACCESSREPORT

Columns

The PROPERTIES_ACCESSREPORT_DIMENSIONHEADERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_ACCESSREPORT _PROPERTYID*	VarChar(64)	References: PROPERTIES_ACCESSREPORT .PROPERTYID
POSITION*	Integer	
DIMENSIONNAME	VarChar(64)	

PROPERTIES_ACCESSREPORT_METRICHEADERS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}:runAccessReport
```

Parent Table

PROPERTIES_ACCESSREPORT

Columns

The PROPERTIES_ACCESSREPORT_METRICHEADERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_ACCESSREPORT _PROPERTYID*	VarChar(64)	References: PROPERTIES_ACCESSREPORT .PROPERTYID
POSITION*	Integer	
METRICNAME	VarChar(64)	

PROPERTIES_ACCESSREPORT_ROWS

Endpoint

`https://analyticsadmin.googleapis.com/v1/properties/{propertyId}:runAccessReport`

Parent Table

PROPERTIES_ACCESSREPORT

Columns

The PROPERTIES_ACCESSREPORT_ROWS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_ACCESSREPORT _PROPERTYID*	VarChar(64)	References: PROPERTIES_ACCESSREPORT .PROPERTYID
POSITION*	Integer	
PROPERTIES_ACCESSREPORT_ROW	VarChar(255)	

PROPERTIES_ACKNOWLEDGEUSERDATACOLLECTION

Endpoint

`https://analyticsadmin.googleapis.com/v1/properties/{propertyId}:acknowledgeUserDataCollection`

Columns

The PROPERTIES_ACKNOWLEDGEUSERDATACOLLECTION table contains the following columns. Columns marked with an asterisk comprise the primary key.

Note: The ACKNOWLEDGEMENT column returns the following string value for all rows:

```
I acknowledge that I have the necessary privacy disclosures and rights from my end users
for the collection and processing of their data, including the association of such data
with the visitation information Google Analytics collects from my site and/or app property.
```

Column Name	Data Type
PROPERTYID	VarChar(64)
ACKNOWLEDGEMENT	VARCHAR(0)

PROPERTIES_ATTRIBUTIONSETTINGS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/attributionSettings
```

Columns

The PROPERTIES_ATTRIBUTIONSETTINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
ACQUISITIONCONVERSIONEVENTLOOKBACKWINDOW	VarChar(40)
OTHERCONVERSIONEVENTLOOKBACKWINDOW	VarChar(40)
PROPERTYID	VarChar(64)
REPORTINGATTRIBUTIONMODEL	VarChar(30)

PROPERTIES_AUDIENCES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/audiences
```

Columns

The PROPERTIES_AUDIENCES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
AUDIENCEID*	VarChar(64)
NAME*	VarChar(64)
ADSPERSONALIZATIONENABLED	Boolean
DESCRIPTION	VarChar(64)
DISPLAYNAME	VarChar(64)
EVENTTRIGGER_EVENTNAME	VarChar(64)
EVENTTRIGGER_LOGCONDITION	VarChar(30)
EXCLUSIONDURATIONMODE	VarChar(40)
MEMBERSHIPDURATIONDAYS	Integer

PROPERTIES_AUDIENCES_FILTERCLAUSES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/audiences
```

Parent Table

PROPERTIES_AUDIENCES

Columns

The PROPERTIES_AUDIENCES_FILTERCLAUSES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_AUDIENCES_PROPERTYID*	VarChar(64)	References: PROPERTIES_AUDIENCES .PROPERTYID
PROPERTIES_AUDIENCES_AUDIENCEID*	VarChar(64)	References: PROPERTIES_AUDIENCES .AUDIENCEID
PROPERTIES_AUDIENCES_NAME*	VarChar(64)	References: PROPERTIES_AUDIENCES.NAME

Column Name	Data Type	Notes
POSITION*	Integer	
CLAUSETYPE	VarChar(30)	
SEQUENCEFILTER_SCOPE	VarChar(40)	
SEQUENCEFILTER _SEQUENCEMAXIMUMDURATION	VarChar(64)	
SIMPLEFILTER_FILTEREXPRESSION	VarChar(255)	
SIMPLEFILTER_SCOPE	VarChar(40)	

PROPERTIES_AUDIENCES_SEQUENCESTEPS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/audiences
```

Parent Table

PROPERTIES_AUDIENCES_FILTERCLAUSES

Columns

The PROPERTIES_AUDIENCES_SEQUENCESTEPS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_AUDIENCES_PROPERTYID*	VarChar(64)	References: PROPERTIES_AUDIENCES .PROPERTYID
PROPERTIES_AUDIENCES_AUDIENCEID*	VarChar(64)	References: PROPERTIES_AUDIENCES .AUDIENCEID
PROPERTIES_AUDIENCES_NAME*	VarChar(64)	References: PROPERTIES_AUDIENCES.NAME
PROPERTIES_AUDIENCES_FILTERCLAUSES _POSITION*	Integer	References: PROPERTIES_AUDIENCES _FILTERCLAUSES.POSITION
POSITION*	Integer	
CONSTRAINTDURATION	VarChar(64)	
FILTEREXPRESSION	VarChar(255)	

Column Name	Data Type	Notes
IMMEDIATELYFOLLOWS	Boolean	
SCOPE	VarChar(40)	

PROPERTIES_CONVERSIONEVENTS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/conversionEvents
```

Columns

The PROPERTIES_CONVERSIONEVENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
EVENTID*	VarChar(64)
NAME*	VarChar(64)
CREATETIME	VarChar(64)
CUSTOM	Boolean
DELETABLE	Boolean
EVENTNAME	VarChar(64)

PROPERTIES_CUSTOMDIMENSIONS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/{propertyId}/customDimensions
```

Columns

The PROPERTIES_CUSTOMDIMENSIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)

Column Name	Data Type
DIMENSIONID*	VarChar(64)
NAME*	VarChar(64)
DESCRIPTION	VarChar(256)
DISALLOWADSPERSONALIZATION	Boolean
DISPLAYNAME	VarChar(64)
PARAMETERNAME	VarChar(64)
SCOPE	VarChar(30)

PROPERTIES_CUSTOMMETRICS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/customMetrics
```

Columns

The PROPERTIES_CUSTOMMETRICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
METRICID*	VarChar(64)
NAME*	VarChar(124)
DISPLAYNAME	VarChar(64)
MEASUREMENTUNIT	VarChar(64)
PARAMETERNAME	VarChar(64)
SCOPE	VarChar(64)

PROPERTIES_CUSTOMMETRICS_RESTRICTEDMETRICTYPE

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/customMetrics
```

Parent Table

PROPERTIES_CUSTOMMETRICS

Columns

The PROPERTIES_CUSTOMMETRICS_RESTRICTEDMETRICTYPE table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_CUSTOMMETRICS _PROPERTYID*	VarChar(64)	References: PROPERTIES_CUSTOMMETRICS .PROPERTYID
PROPERTIES_CUSTOMMETRICS_METRICID*	VarChar(64)	References: PROPERTIES_CUSTOMMETRICS .METRICID
PROPERTIES_CUSTOMMETRICS_NAME*	VarChar(124)	References: PROPERTIES_CUSTOMMETRICS .NAME
POSITION*	Integer	
PROPERTIES_CUSTOMMETRICS _RESTRICTEDMETRICTYPE	VarChar(30)	

PROPERTIES_DATARETENTIONSETTINGS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/dataRetentionSettings
```

Columns

The PROPERTIES_DATARETENTIONSETTINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
EVENTDATARETENTION	VarChar(30)

Column Name	Data Type
PROPERTYID	VarChar(64)
RESETUSERDATAONNEWACTIVITY	Boolean

PROPERTIES_DATASTREAMS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/dataStreams
```

Columns

The PROPERTIES_DATASTREAMS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
DATASTREAMID*	VarChar(64)
NAME*	VarChar(64)
ANDROIDAPPSTREAMDATA_FIREBASEAPPID	VarChar(64)
ANDROIDAPPSTREAMDATA_PACKAGENAME	VarChar(64)
CREATETIME	Timestamp(9)
DISPLAYNAME	VarChar(64)
IOSAPPSTREAMDATA_BUNDLEID	VarChar(64)
IOSAPPSTREAMDATA_FIREBASEAPPID	VarChar(64)
TYPE	VarChar(64)
UPDATETIME	Timestamp(9)
WEBSTREAMDATA_DEFAULTURI	VarChar(64)
WEBSTREAMDATA_MEASUREMENTID	VarChar(64)

PROPERTIES_DATASTREAMS_GLOBALSITETAG

Endpoint

https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/dataStreams/{dataStreamId}/globalSiteTag

Columns

The PROPERTIES_DATASTREAMS_GLOBALSITETAG table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
DATASTREAMID	VarChar(64)
NAME	VarChar(64)
SNIPPET	VarChar(64)

PROPERTIES_DATASTREAMS_MEASUREMENTPROTOCOLSECRETS

Endpoint

https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/dataStreams/{dataStreamId}/measurementProtocolSecrets

Columns

The PROPERTIES_DATASTREAMS_MEASUREMENTPROTOCOLSECRETS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
DATASTREAMID*	VarChar(64)
SECRETSID*	VarChar(64)
NAME*	VarChar(64)
DISPLAYNAME	VarChar(64)
SECRETVALUE	VarChar(64)

PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKS

Endpoint

https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/displayVideo360AdvertiserLinks

Columns

The PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
LINKID*	VarChar(64)
NAME*	VarChar(64)
ADSPERSONALIZATIONENABLED	Boolean
ADVERTISERDISPLAYNAME	VarChar(64)
ADVERTISERID	VarChar(64)
CAMPAIGNDATASHARINGENABLED	Boolean
COSTDATASHARINGENABLED	Boolean

PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKPROPOSALS

Endpoint

https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/displayVideo360AdvertiserLinkProposals

Columns

The PROPERTIES_DISPLAYVIDEO360ADVERTISERLINKPROPOSALS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
PROPOSALID*	VarChar(64)
NAME*	VarChar(64)
ADSPERSONALIZATIONENABLED	Boolean

Column Name	Data Type
ADVERTISERDISPLAYNAME	VarChar(64)
ADVERTISERID	VarChar(64)
CAMPAIGNDATASHARINGENABLED	Boolean
COSTDATASHARINGENABLED	Boolean
LINKPROPOSALSTATUSDETAILS_LINKPROPOSALINITIATINGPRODUCT	VarChar(64)
LINKPROPOSALSTATUSDETAILS_LINKPROPOSALSTATE	VarChar(40)
LINKPROPOSALSTATUSDETAILS_REQUESTOREMAIL	VarChar(64)
VALIDATIONEMAIL	VarChar(64)

PROPERTIES_FIREBASELINKS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/firebaseLinks
```

Columns

The PROPERTIES_FIREBASELINKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(67)
CREATETIME	Timestamp(3)
FIREBASEID	VarChar(64)
PROJECT	VarChar(64)
PROPERTYID	VarChar(64)

PROPERTIES_GOOGLEADSLINKS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/googleAdsLinks
```

Columns

The PROPERTIES_GOOGLEADSLINKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(69)
ADSPERSONALIZATIONENABLED	Boolean
CREATETIME	Timestamp(9)
CREATOREMAILADDRESS	VarChar(64)
CUSTOMERID	BigInt
GOOGLEADSID	VarChar(64)
PROPERTYID	VarChar(64)
UPDATETIME	Timestamp(9)

PROPERTIES_GOOGLESIGNALSSETTINGS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/googleSignalsSettings
```

Columns

The PROPERTIES_GOOGLESIGNALSSETTINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
NAME*	VarChar(64)
CONSENT	VarChar(30)
PROPERTYID	VarChar(64)
STATE	VarChar(30)

PROPERTIES_USERLINKS

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/userLinks
```

Columns

The PROPERTIES_USERLINKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PROPERTYID*	VarChar(64)
USERLINKID*	VarChar(64)
NAME*	VarChar(64)
EMAILADDRESS	VarChar(64)

PROPERTIES_USERLINKS_AUDIT

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/userLinks:audit
```

Columns

The PROPERTIES_USERLINKS_AUDIT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ROWID*	VarChar(32)
EMAILADDRESS	VarChar(64)
NAME	VarChar(64)
PROPERTYID	VarChar(64)

PROPERTIES_USERLINKS_AUDIT_DIRECTROLES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/userLinks:audit
```

Parent Table

PROPERTIES_USERLINKS_AUDIT

Columns

The PROPERTIES_USERLINKS_AUDIT_DIRECTROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_USERLINKS_AUDIT_ROWID*	VarChar(32)	References: PROPERTIES_USERLINKS_AUDIT .ROWID
POSITION*	Integer	
PROPERTIES_USERLINKS_AUDIT _DIRECTROLE	VarChar(64)	
PROPERTYID	VarChar(64)	

PROPERTIES_USERLINKS_AUDIT_EFFECTIVEROLES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/userLinks:audit
```

Parent Table

PROPERTIES_USERLINKS_AUDIT

Columns

The PROPERTIES_USERLINKS_AUDIT_EFFECTIVEROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_USERLINKS_AUDIT_ROWID*	VarChar(32)	References: PROPERTIES_USERLINKS_AUDIT .ROWID
POSITION*	Integer	
PROPERTIES_USERLINKS_AUDIT _EFFECTIVEROLE	VarChar(64)	
PROPERTYID	VarChar(64)	

PROPERTIES_USERLINKS_DIRECTROLES

Endpoint

```
https://analyticsadmin.googleapis.com/v1/properties/{propertyId}/userLinks
```

Parent Table

PROPERTIES_USERLINKS

Columns

The PROPERTIES_USERLINKS_DIRECTROLES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
PROPERTIES_USERLINKS_PROPERTYID*	VarChar(64)	References: PROPERTIES_USERLINKS .PROPERTYID
PROPERTIES_USERLINKS_USERLINKID*	VarChar(64)	References: PROPERTIES_USERLINKS .USERLINKID
PROPERTIES_USERLINKS_NAME*	VarChar(64)	References: PROPERTIES_USERLINKS.NAME
POSITION*	Integer	
PROPERTIES_USERLINKS_DIRECTROLE	VarChar(64)	

PROPERTIESLIST

Endpoint

`https://analyticsadmin.googleapis.com/v1/properties`

Columns

The PROPERTIESLIST table contains the following columns. Columns marked with an asterisk comprise the primary key.

Note: When querying the PROPERTIESLIST table, you must filter by the PROPERTYID to return results; otherwise, the service returns a `400 Bad Request` error. For example:

```
SELECT * FROM PROPERTIESLIST WHERE FILTER = 'parent:properties/12345679' LIMIT 10
```

Column Name	Data Type
NAME*	VarChar(64)
ACCOUNT	VarChar(64)
CREATETIME	VarChar(64)
CURRENCYCODE	VarChar(64)
DELETETIME	VarChar(64)

Column Name	Data Type
DISPLAYNAME	VarChar(64)
EXPIRETIME	VarChar(64)
FILTER	VarChar(64)
INDUSTRYCATEGORY	VarChar(35)
PARENT	VarChar(64)
PROPERTYID	VarChar(64)
PROPERTYTYPE	VarChar(23)
SERVICELEVEL	VarChar(25)
SHOWDELETED	Boolean
TIMEZONE	VarChar(64)
UPDATETIME	VarChar(64)

REALTIMEREPORTS

Endpoint

`<hostname>/v1/properties/{propertyId}:runRealtimeReport`

Columns

The REALTIMEREPORTS view contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ROWID*	VarChar(32)	
_APPVERSION	VarChar(32767)	dimension
_AUDIENCEID	VarChar(32767)	dimension
_AUDIENCENAME	VarChar(32767)	dimension
_CITY	VarChar(32767)	dimension
_CITYID	VarChar(32767)	dimension
_COUNTRY	VarChar(32767)	dimension

Column Name	Data Type	Notes
_COUNTRYID	VarChar(32767)	dimension
_DEVICECATEGORY	VarChar(32767)	dimension
_EVENTNAME	VarChar(32767)	dimension
_MINUTESAGO	VarChar(32767)	dimension
_PLATFORM	VarChar(32767)	dimension
_STREAMID	VarChar(32767)	dimension
_STREAMNAME	VarChar(32767)	dimension
_UNIFIEDSCREENNAME	VarChar(32767)	dimension
ACTIVEUSERS	Integer	metric
CONVERSIONS	Integer	metric
ENDMINUTESAGO	INTEGER	
EVENTCOUNT	Integer	metric
LIMIT	INTEGER	
PROPERTYID	Integer	
PROPERTYQUOTA _CONCURRENTREQUESTS_CONSUMED	Integer	
PROPERTYQUOTA _CONCURRENTREQUESTS_REMAINING	Integer	
PROPERTYQUOTA _POTENTIALLYTHRESHOLD EDREQUESTSPERHOUR_CONSUMED	Integer	
PROPERTYQUOTA _POTENTIALLYTHRESHOLD EDREQUESTSPERHOUR_REMAINING	Integer	
PROPERTYQUOTA _SERVERERRORSPERPROJECTPERHOUR _CONSUMED	Integer	
PROPERTYQUOTA _SERVERERRORSPERPROJECTPERHOUR _REMAINING	Integer	

Column Name	Data Type	Notes
PROPERTYQUOTA_TOKENSPERDAY_CONSUMED	Integer	
PROPERTYQUOTA_TOKENSPERDAY_REMAINING	Integer	
PROPERTYQUOTA_TOKENSPERHOUR_CONSUMED	Integer	
PROPERTYQUOTA_TOKENSPERHOUR_REMAINING	Integer	
PROPERTYQUOTA_TOKENSPERPROJECTPERHOUR_CONSUMED	Integer	
PROPERTYQUOTA_TOKENSPERPROJECTPERHOUR_REMAINING	Integer	
RETURNPROPERTYQUOTA	BOOLEAN	
SCREENPAGEVIEWS	Integer	metric
STARTMINUTESAGO	INTEGER	