Progress DataDirect for JDBC for Google Analytics 4

User's Guide

Release 6.0.0

Copyright

Table of Contents

1

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

The Progress® DataDirect® for JDBC™ for Google Analytics™ 4 driver supports SQL read-only access for JDBC 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.

In addition, the driver employs a SQL engine component that provides support to SQL constructs to enable the most comprehensive SQL support and JDBC standard connectivity. To support SQL access to Google Analytics services, the driver creates a relational map of the Google Analytics 4 data model and translates SQL statements to Google Analytics REST API requests.

For an overview of how the driver exposes the Google Analytics 4 data model as a relational schema, and how to query Google Analytics, see Introduction to the Google Analytics Data Model.

The documentation for the driver also includes the *Progress DataDirect for JDBC Drivers Reference*. The reference provides general reference information for all DataDirect drivers for JDBC, including content on troubleshooting, supported SQL escapes, and DataDirect tools.

For the complete documentation set, visit the Progress DataDirect Connectors Documentation Hub: https://docs.progress.com/category/datadirect-google-analytics.

For details, see the following topics:

- What's new in this release?
- Requirements
- Installing and setting up the driver

- Driver and DataSource classes
- Connection URL examples
- Data types
- SQL escape sequences
- DataDirect tools
- Troubleshooting
- Additional information
- Contacting Technical Support

What's new in this release?

Support and certification

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

- Release Notes: https://www.progress.com/datadirect-connectors/whats-new#jdbc
- DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf

Changes Since 6.0.0 GA

- **Enhancements**
  - The driver has been enhanced to comply with FIPS standards for data encryption. As part of this enhancement, the driver was tested with FIPS 140-3 enabled using a Red Hat OpenJDK 21 on a Red Hat Universal Base Image 9 instance. See FIPS (Federal Information Processing Standard) on page 49 for details.
- **Changed behavior**
  - The connection property `SpyAttributes` has been updated to exclude the attribute `load=classname`, which was previously used to load the driver specified by the given class name. See SpyAttributes on page 71 for details.

Highlights of 6.0.0 Release

- The driver supports SQL read-only access to Google Analytics using Google Analytics 4 (GA4) Data and Admin API. See Supported SQL statements and extensions on page 79 for details.
- The driver supports JDBC core functions. For details, refer to "JDBC support" in the *Progress DataDirect for JDBC Drivers Reference*.
- The driver supports Google Analytics data types. See Data types on page 14 and getTypeInfo() on page 15 for details.
- The driver supports OAuth 2.0 authentication. See OAuth 2.0 Authentication on page 43 for further details.
- The driver supports the handling of large result sets with paging, and the FetchSize on page 63 connection property.

- The driver includes the Progress DataDirect Google Analytics 4 Configuration Manager for quick configuration and testing of your driver in a web browser. The tool allows you to:
  - Generate and edit connection URLs
  - Test connect your connection URLs
  - Add tables to a relational view of Google Analytics data
  - Execute SQL commands for testing

  For details, see Generating connection URLs with the Configuration Manager on page 36 and Testing connections and queries on page 37.

Requirements

The driver is compatible with JDBC 2.0, 3.0, and 4.0.

The driver requires a Java Virtual Machine (JVM) that is Java SE 8 or higher, including Oracle JDK, OpenJDK, and IBM SDK (Java) distributions.

Installing and setting up the driver

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.
      - **Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.exe`
      - **Non-Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.jar`
   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 JDBC Drivers Installation Guide*.
2. Set your system CLASSPATH to include the driver `.jar` file. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. The following examples demonstrate setting the CLASSPATH from a command line using the default installation directory.
   - **Windows Example**

     ```
     CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics4.jar
     ```
   - **UNIX/LINUX Example**

     ```
     CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/googleanalytics4.jar
     ```
3. Configure your driver using one of the following methods:

- **Connection URL**: You can begin using the driver immediately by passing a connection URL with your application or tool. The following example shows you the minimum properties required to connect using OAuth 2.0 authentication with a refresh token.

  ```
  jdbc:datadirect:googleanalytics4: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;
  ```

  You can also generate a connection string using the Progress DataDirect Google Analytics 4 Configuration Manager. For details, see Generating connection URLs with the Configuration Manager.

  Important: Although you can still connect to Google Analytics, you cannot query your data without defining the tables in your schema using the AddTables property. You can generate a value for this property using the **Configure Logical Schema** button in the Configuration Manager and copy it into your string. See Generating connection URLs with the Configuration Manager for details.

  Note: You can fetch tokens using the Configuration Manager. See OAuth 2.0 Authentication for more information.

- **Data sources**: The driver also supports connecting using JDBC data sources. A JDBC data source is a Java object, specifically a DataSource object, that defines connection information required for a JDBC driver to connect to the database. See Connecting using data sources for more information.

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

4. Set the values for any optional properties that you want to configure. For additional information on optional features and functionality, see the following resources:

   - Connection URL 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.
   - Connection property descriptions provides a complete list of supported properties by functionality.
   - Performance considerations describes connection properties that affect performance, along with recommended settings.

5. Connect to your service 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:

   - Progress DataDirect Google Analytics 4 Configuration Manager: The Google Analytics 4 Configuration Manager is a browser-based tool that allows you to quickly generate connection URLs, test connections, and execute test queries.
   - DataDirect Test: DataDirect Test allows you to test connect, execute SQL statements, and practice using the JDBC API right out of the box.
   - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
   - DbVisualizer: DB Visualizer is a database tool that allows you to connect and execute SQL statements against your data.

- Supported SQL statements and extensions: 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.

Driver and DataSource classes

The following are the `Driver` and `DataSource` classes used by the driver:

Driver class:

com.ddtek.jdbc.googleanalytics4.GoogleAnalytics4Driver

DataSource class:

com.ddtek.jdbcx.googleanalytics4.GoogleAnalytics4DataSource

Connection URL examples

After setting the CLASSPATH, the connection information needs to be passed in the form of a connection URL. 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.

Note:

- You can also use the DataDirect Configuration Manager tool to generate and test connection URLs. For more information, see "Generating connection URLs with the Configuration Manager."
- Connection property names are case-insensitive. For example, `AuthenticationMethod` is the same as `authenticationmethod`.
- For connection properties that support string values, use the following escape sequence to specify values containing leading or trailing spaces and curly brackets: {*value*}. For example: `ProxyPassword={hello }` or `ProxyPassword={{hello}}`.

Important: Although you can still connect to Google Analytics, you cannot query your data without defining the tables in your schema using the AddTables property. You can generate a value for this property using the **Configure Logical Schema** button in the Configuration Manager and copy it into your string. See Generating connection URLs with the Configuration Manager for details.

- OAuth 2 authentication example
- Proxy server example

OAuth 2 Authentication example

This string includes the properties used to connect with OAuth 2 authentication.

```
jdbc:datadirect:googleanalytics4: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;[property=value[;...]];
```

For more information on these properties and values, see OAuth 2.0 Authentication on page 43.

Proxy Server

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

```
jdbc:datadirect:googleanalytics4:ProxyHost=pserver;ProxyPassword=secret;
ProxyPort=808;ProxyUser=proxy_user;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;[property=value[;...]];
```

For more information on these properties and values, see Table 5: Proxy Server Properties on page 53.

See also

Generating connection URLs with the Configuration Manager on page 36

Data types

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

See "getTypeInfo()" for getTypeInfo() results of data types supported by the driver.

Table 1: Google Analytics 4 Data Types

Google Analytics 4 Data Type	JDBC Data Type
BigInt	BIGINT
Binary	BINARY
Bit	BIT
Boolean	BOOLEAN
Char	CHAR
Date	DATE
Decimal	DECIMAL
Double	DOUBLE
Float	FLOAT
GUID	CHAR

Google Analytics 4 Data Type	JDBC Data Type
Integer	INTEGER
JSON	VARCHAR
LongVarBinary	LONGVARBINARY
LongVarChar	LONGVARCHAR
NVarChar	NVARCHAR
SmallInt	SMALLINT
Time	TIME
Timestamp	TIMESTAMP
TinyInt	TINYINT
VarBinary	VARBINARY
VarChar	VARCHAR

getTypeInfo()

The DatabaseMetaData.getTypeInfo() method returns information about data types. The following table provides getTypeInfo() results for supported data types.

Table 2: getTypeInfo() Results

TYPE_NAME = BigInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -5 (BIGINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = BigInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 10 PRECISION = 19 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = Binary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -2 (BINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Binary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Bit AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -7 (BIT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Bit MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Boolean AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 16 (BOOLEAN) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Boolean MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = Char AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 1 (CHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Char MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Date AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 91 (DATE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = DATE ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Date MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 10 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Decimal AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 3 (DECIMAL) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Decimal MAXIMUM_SCALE = 1000	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 10 PRECISION = 1000 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = Double AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Double MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 2 PRECISION = 53 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = Float AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 6 (FLOAT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Float MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 2 PRECISION = 24 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = GUID AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 1 (CHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = GUID MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 36 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = Integer AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 4 (INTEGER) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Integer MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 10 PRECISION = 10 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = JSON AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = JSON MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = LongVarBinary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -4 (LONGVARBINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = LongVarBinary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 0 (**typePredNone**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = LongVarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = -1 (LONGVARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = LongVarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 0 (**typePredNone**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = NVarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = -9 (NVARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = NVarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = SmallInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 5 (SMALLINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = SmallInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 10 PRECISION = 5 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = Time AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 92 (TIME) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = TIME ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Time MAXIMUM_SCALE = 9	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 12 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Timestamp AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 93 (TIMESTAMP) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = TIMESTAMP ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Timestamp MAXIMUM_SCALE = 9	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 23 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = TinyInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -6 (TINYINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = TinyInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = 10 PRECISION = 3 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = VarBinary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -3 (VARBINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = VarBinary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = VarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = VarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 (**typeNullable**) NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 (**typeSearchable**) SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

SQL escape sequences

The driver supports the following SQL escape sequences.

- Date, Time, and Timestamp Escape Sequences
- Scalar Functions
- Outer Join Escape Sequences
- LIKE Escape Character Sequence for Wildcards

Refer to "SQL escape sequences" in the *Progress DataDirect for JDBC Drivers Reference* for information about SQL escape sequences.

Supported scalar functions

The driver supports the scalar functions in the following table. Note that your database system may not support all these functions. Refer to the documentation for your database system to find out which functions are supported by your database.

In addition, you can also determine the supported scalar functions by using DatabaseMetaData methods.

You can use scalar functions in SQL statements with the following syntax:

`{fn` *`scalar-function`*`}`

where:

`scalar-function`

is a scalar function supported by the drivers, as listed in the following table.

Example:

```
SELECT id, name FROM emp WHERE name LIKE {fn UCASE('Smith')}
```

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

Table 3: Supported Scalar Functions

String Functions	Numeric Functions	Timedate Functions	System Functions
ASCII	ABS	CURDATE	CURSESSIONID
BIT_LENGTH	ACOS	CURRENT_DATE	DATABASE
CHAR	ASIN	CURRENT_TIME	IDENTITY
CHAR_LENGTH	ATAN	CURRENT_TIMESTAMP	USER
CHARACTER_LENGTH	ATAN2	CURTIME	
CONCAT	BITAND	DATEDIFF	
DIFFERENCE	BITOR	DATE_ADD	
HEXTORAW	BITXOR	DATE_SUB	
INSERT	CEILING	DAY	
LCASE	COS	DAYNAME	
LEFT	COT	DAYOFMONTH	
LENGTH	DEGREES	DAYOFWEEK	
LOCATE	EXP	DAYOFYEAR	
LOCATE_2	FLOOR	EXTRACT	

String Functions	Numeric Functions	Timedate Functions	System Functions
LOWER	LOG	HOUR	
LTRIM	LOG10	MINUTE	
OCTET_LENGTH	MOD	MONTH	
RAWTOHEX	PI	MONTHNAME	
REPEAT	POWER	NOW	
REPLACE	RADIANS	QUARTER	
RIGHT	RAND	SECOND	
RTRIM	ROUND	SECONDS_SINCE_MIDNIGHT	
SOUNDEX	ROUNDMAGIC	TIMESTAMPADD	
SPACE	SIGN	TIMESTAMPDIFF	
SUBSTR	SIN	TO_CHAR	
SUBSTRING	SQRT	WEEK	
UCASE	TAN	YEAR	
UPPER	TRUNCATE		

DataDirect tools

Progress DataDirect for JDBC drivers install the set of tools described in this section. For detailed instructions on using these tools, refer to the corresponding topics in the *Progress DataDirect for JDBC Drivers Reference*.

- DataDirect Test allows you to test your JDBC driver and learn the JDBC API.
- DataDirect Connection Pool Manager allows you to pool connections when accessing databases. When your applications use connection pooling, connections are reused rather than created each time a connection is requested. Because establishing a connection is among the most costly operations an application may perform, using Connection Pool Manager to implement connection pooling can significantly improve performance.
- Statement Pool Monitor loads statements into and remove statements from the statement pool as well as generate information to help you troubleshoot statement pooling performance.
- DataDirect Spy logs detailed information about calls your driver makes that can be used for troubleshooting.

Troubleshooting

The *Progress DataDirect for JDBC Drivers Reference* provides information on troubleshooting problems should they occur. Refer to the "Troubleshooting" section in the *Reference* for details.

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 JDBC Drivers Reference* or use the links below to view some common topics:

- "JDBC support" describes support for JDBC interfaces and methods for the Progress DataDirect for JDBC drivers.
- "JDBC extensions" describes the JDBC extensions provided by the com.ddtek.jdbc.extensions package.
- "SQL escape sequences for JDBC" provides an overview of SQL escape sequences for JDBC. In addition, it documents the scalar functions that you use in SQL statements.
- "Security best practices for JDBC applications" describes the security best practices you should employ when developing and deploying your application with the driver.

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.

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."

For details, see the following topics:

- Interactive SQL
- Tableau
- DbVisualizer

Interactive SQL

Note: Before you start, you will need to know the value of the AddTables property that is used to define your schema. You will still be able to connect without specifying a value, but you will not be able to query your data. You can generate a value for the property using the **Configure Logical Schema** tool in the Configuration Manager. See Generating connection URLs with the Configuration Manager for details.

After you have installed your driver, you can use the driver to access your data with the Interactive SQL tool. Interactive SQL supports a command line interface that allows you to connect to a data source, execute SQL statements and retrieve results for display on a terminal.

To execute commands with Interactive SQL:

1. Start the ISQL tool. From a command line, enter the following:

   ```
   java -jar googleanalytics4.jar --isql
   ```

2. Enter connection properties one at a time by typing *property=value*, then pressing **Enter**. For example, to configure the ClientID property:

   ```
   ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com
   ```

3. After specifying values for your properties, type `connect`, then press **Enter**. If successful, the tool will return a confirmation message.

   Note: If you are unable to connect, you can review the URL by entering the `SHOW URL` command.

4. At the `ISQL>` prompt, issue a SQL command to query or modify the data source; then, press **Enter**. For example:

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

   Note: SQL commands must be terminated by a semi-colon.

   Note: In addition to SQL commands, the tool supports a set of proprietary commands. Type `Help` at the prompt for a list of supported commands and syntax.

   The results of the command are displayed in the terminal.

5. After you are finished executing queries and commands, you can disconnect from the data source by typing the following; then, pressing **Enter**:

   ```
   DISCONNECT
   ```

6. To end the session, type `exit`; then, press **ENTER**.

Tableau

Note: Before you start, you will need to know the value of the AddTables property that is used to define your schema. You will still be able to connect without specifying a value, but you will not be able to query your data. You can generate a value for the property using the **Configure Logical Schema** tool in the Configuration Manager. See Generating connection URLs with the Configuration Manager for details.

After you have installed your driver and defined it on the CLASSPATH, 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 `\lib\xx` subdirectory of the Progress DataDirect installation directory; then, locate the `jar` file for your driver:

   ```
   googleanalytics4.jar
   ```

2. Copy the `.jar` file for your driver into the following directory:

   Windows: `C:\Program Files\Tableau\Drivers`

   Linux: `/opt/tableau/tableau_driver/jdbc`

3. Open Tableau. From the **Connect** menu, select **Other Databases (JDBC)**.
4. In the **Other Databases (JDBC)** dialog, provide values for the following fields; then, click **Sign In**.

   - **URL**: Copy and paste your connection URL into this field. The following examples show how to connect using OAuth 2 authentication with a refresh token.

     ```
     jdbc:datadirect:googleanalytics4:AddTables='{myTableDefinitionString}';
     ClientID=client_id.apps.googleusercontent.com;
     ClientSecret=client_secret;RefreshToken=refresh_token;
     ```

     Note: See OAuth 2.0 Authentication on page 43 for details.

   - **Dialect**: Select **SQL92** (the default) from the drop-down box.

5. The **Data Source** window appears. In the **Schema** field, select the schema for the service you want to use.
6. In the **Table** field, the tables stored in the selected schema are now exposed and available for selection.

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

DbVisualizer

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with the third-party DbVisualizer tool. The following topics guide you through using DbVisualizer to add your driver, connect, and execute SQL statements.

Adding a driver

Note: Before you start, you will need to know the value of the AddTables property that is used to define your schema. You will still be able to connect without specifying a value, but you will not be able to query your data. You can generate a value for the property using the **Configure Logical Schema** tool in the Configuration Manager. See Generating connection URLs with the Configuration Manager for details.

To add a driver with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Tools**>**Driver Manager**. The Driver Manager window opens.
3. From the Driver Manager menu, select **Driver**>**Create Driver**.

4. Click the  button to navigate to the location of the driver jar file; then, click **OK**. The following are the default locations for the driver:

   Windows

   ```
   C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics4.jar
   ```

   Linux

   ```
   /opt/Progress/DataDirect/JDBC/lib/60/googleanalytics4.jar
   ```

5. Provide values for the following fields; then, close the Driver Manager window.

   - **Name**: Type an alias for your driver. For example:

     ```
     Google Analytics 4
     ```

   - **URL Format**: Optionally, specify the format of the connection string for your driver. For example:

     ```
     jdbc:datadirect:googleanalytics4:DefaultQueryOptions=startDate=20daysAgo;currencyCode:USD
     ```

   - **Driver Class**: From the drop-down menu, select the driver class for your driver:

     ```
     com.ddtek.jdbc.googleanalytics4.GoogleAnalytics4Driver
     ```

You can now use your driver with DbVisualizer. Proceed to "Connecting and executing SQL statements" for information on connecting and executing SQL statements.

Connecting and executing SQL statements

To use the driver to access data with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Database**>**New Connection**. When prompted to use the Connection Wizard, click **OK**.
3. Provide the following information when prompted; then, click **Next** to proceed:

   - **Connection alias**: Type the name to be used when referring to this connection.
   - **Driver**: Select the alias that you provided for your driver from the drop-down menu.

4. Provide values for the following fields; then, click **Finish**.

   - **Database URL**: Copy and paste your connection URL into this field. The following examples show how to connect using basic authentication.

     Note: You can also generate connection strings using Google Analytics 4 Configuration Manager. For more information, see Generating connection URLs with the Configuration Manager on page 36.

     ```
     jdbc:datadirect:googleanalytics4:ClientID=client_id.apps.googleusercontent.com;
          ClientSecret=client_secret;RefreshToken=refresh_token;
     ```

Note: See OAuth 2.0 Authentication on page 43 for details.

5. To execute SQL statements, select **SQL Commander>New SQL Commander**. A SQL Commander tab opens.
6. Select values for the following fields:
   - **Database Connection**: Select connection alias you provided for the connection from the drop-down menu.
   - **Schema**: Select the schema you want to execute queries against from the drop-down menu.
7. In the SQL Commander tab, enter SQL commands you want to execute; then select **SQL Commander>Execute**. For example:

   To select all of the rows from the A_CUSTOMER table:

   ```
   SELECT * FROM A_CUSTOMER
   ```

   To select the URLs for a specified issue :

   ```
   SELECT CUSTOMER FROM <SCHEMA_NAME>.A_CUSTOMER WHERE CUSTOMERNAME = <customer_name>
   ```

   See "Supported SQL statements and extensions" for the supported syntax used to execute SQL statements.

   Note: If you are fetching large sets of data, you may want to limit the results using the Max Rows and Max Chars fields.

You have successfully accessed your data with DbVisualizer.

See also

Supported SQL statements and extensions on page 79

3

Configuring and connecting

This section provides information on how to connect to your data store using either the JDBC Driver Manager or DataDirect JDBC data sources, as well as information on how to implement and use functionality supported by the driver.

After the driver has been installed and defined on your classpath, you can connect from your application to your data in either of the following ways.

- Using the JDBC `DriverManager` by specifying the connection URL in the `DriverManager.getConnection()` method.
- Creating a JDBC data source that can be accessed through the Java Naming Directory Interface (JNDI).

For details, see the following topics:

Setting the classpath

The driver must be defined on your CLASSPATH before you can connect. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. If the driver is not defined on your CLASSPATH, you will receive a `class not found` exception when trying to load the driver. Set your system CLASSPATH to include the driver jar file as shown, where *`install_dir`* is the path to your product installation directory.

```
install_dir/lib/60/googleanalytics4.jar
```

Windows Example

```
CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\googleanalytics4.jar
```

UNIX Example

```
CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/googleanalytics4.jar
```

Connecting using the JDBC Driver Manager

One way to connect to a service is through the JDBC DriverManager using the `DriverManager.getConnection()` method. As the following example shows, this method specifies a string containing a connection URL.

```
Connection conn = DriverManager.getConnection
  (jdbc:datadirect:googleanalytics4:AddTables='{myTableDefinitionString}';
     ClientID=client_id.apps.googleusercontent.com;ClientSecret=client_secret;
     RefreshToken=refresh_token;[property=value[;...]]);
```

Passing the connection URL

After setting the CLASSPATH, the required connection information needs to be passed in the form of a connection URL. The following example includes the properties required for connecting with OAuth 2.0 authentication with a refresh token.

Connection URL Syntax

The connection URL takes the following form:

```
jdbc:datadirect:googleanalytics4:AddTables='{myTableDefinitionString}';
ClientID=client_id.apps.googleusercontent.com;ClientSecret=client_secret;
RefreshToken=refresh_token;Scope=scope;[property=value[;...]];
```

where:

myTableDefinitionString

Specifies a JSON string that defines custom tables, including dimensions and metrics, that appear in the Google Analytics 4 schema. For example:
`'{"TestTable":["sessions","totalUsers","_browser","_sessionSource"]}'`.

Note: You can generate a value for the property using the **Configure Logical Schema** tool in the Configuration Manager. See Generating connection URLs with the Configuration Manager for details.

client_id

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

client_secret

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

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

refresh_token

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 service. To prevent unauthorized access, this value must be securely maintained.

scope

(Optional) specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token. The default is:

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

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

Connection URL Examples:

```
Connection conn = DriverManager.getConnection
 (jdbc:datadirect:googleanalytics4:
  AddTables='{"TestTable":["sessions","totalUsers","_browser","_sessionSource"]}';
  ClientID=ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com;
  ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
  RefreshToken=1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;Scope=scope;
  [property=value[;...]]);
```

See also

Connection property descriptions on page 51

Connection URL examples on page 13

Generating connection URLs with the Configuration Manager

The driver includes a browser-based tool, Progress DataDirect Google Analytics 4 Configuration Manager, that allows you to generate connection URLs, test connections, and execute test queries. This section will guide you through generating and testing a connection URL that can be used by your application.

To generate a connection URL:

1. Open the Google Analytics 4 Configuration Manager by double-clicking the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -jar googleanalytics4.jar
   ```

   The Google Analytics 4 Configuration Manager opens in your default web browser.

2. From the browser window, provide values of the connection properties you want to configure in the corresponding fields. A connection URL will generate in the Connection String field as you provide settings. To view more properties, select the tabs at the top of the page. See "Connection URL examples" for a list of required properties and properties used for different configurations.

   Note: If you do not specify a value for an optional property, the property will be omitted from the string and the default value will be used.

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

   For more information about custom tables, 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 AddTables property 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 **DefaultQueryOptions** field is populated with parameters used for WHERE clause filtering.

4. Optionally, you can manually edit your string by clicking the Edit button ().

5. At any point during the process, you can click **Test Connect** to attempt to connect to the service using the string generated in the Connection String field. In the **Test Connection** window:

a) Provide values for any fields required by your service.

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.

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

6. To use your string, click the Copy button () and paste the string to a location that can be used by your application.
7. Optionally, click **Save** to store your connection string for later use.

See also

Connection URL examples on page 13

Testing connections and queries

You can quickly test a connection string 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 double-clicking on the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -jar googleanalytics4.jar
   ```

   The Google Analytics 4 Configuration Manager opens in your default web browser.

2. Provide a connection string to test using one of the following methods:

   - Entering a string: Click the Edit button (); then, paste your string into the Connection String field. If you prefer, you can also type a string directly into this field.
   - Generating a string: Provide values of the connection properties you want to configure into the corresponding fields. The Google Analytics 4 Configuration Manager will generate a string in the Connection String field based on the values you specify.

3. Click **Test Connect** to attempt to connect to the service using the string specified in the Connection String field. The **Test Connection** window appears.
4. Provide connection property values for any fields required by your service.

5. 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.

Connecting using data sources

A *JDBC data source* is a Java object, specifically a `DataSource` object, that defines connection information required for a JDBC driver to connect to the database. Each JDBC driver vendor provides their own data source implementation for this purpose. A Progress DataDirect data source is Progress DataDirect's implementation of a `DataSource` object that provides the connection information needed for the driver to connect to a database.

Because data sources work with the Java Naming Directory Interface (JNDI) naming service, data sources can be created and managed separately from the applications that use them. Because the connection information is defined outside of the application, the effort to reconfigure your infrastructure when a change is made is minimized. For example, if the database is moved to another database server, the administrator need only change the relevant properties of the `DataSource` object. The applications using the database do not need to change because they only refer to the name of the data source.

How data sources are implemented

Data sources are implemented through a `DataSource` class. A data source class implements the following interfaces.

- `javax.sql.DataSource`
- `javax.sql.ConnectionPoolDataSource` (allows applications to use connection pooling)

Refer to "Connection Pool Manager" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

See also

Driver and DataSource classes on page 13

Creating data sources

The following example files provide details on creating and using Progress DataDirect data sources with the Java Naming Directory Interface (JNDI), where *`install_dir`* is the product installation directory.

- *`install_dir`*`/Examples/JNDI/JNDI_LDAP_Example.java` can be used to create a JDBC data source and save it in your LDAP directory using the JNDI Provider for LDAP.
- *`install_dir`*`/Examples/JNDI/JNDI_FILESYSTEM_Example.java` can be used to create a JDBC data source and save it in your local file system using the File System JNDI Provider.

See "Example data source" for an example data source definition for the example files.

To connect using a JNDI data source, the driver needs to access a JNDI data store to persist the data source information. For a JNDI file system implementation, you must download the File System Service Provider from the Oracle Technology Network Java SE Support downloads page, unzip the files to an appropriate location, and add the `fscontext.jar` and `providerutil.jar` files to your CLASSPATH. These steps are not required for LDAP implementations because the LDAP Service Provider is included with supported versions of Java SE.

See also

Example data source on page 39

Example data source

To configure a data source using the example files, you will need to create a data source definition. The content required to create a data source definition is divided into three sections.

First, you will need to import the data source class. For example:

```
import com.ddtek.jdbcx.googleanalytics4.GoogleAnalytics4DataSource;
```

Next, you will need to set the values and define the data source. For example, the following definition contains the minimum properties required for a connection using the basic authentication method.

Note: Setting the confidential values using a data source is generally not recommended. The data source persists all properties, including properties with confidential values, in clear text.

Note: In a JDBC data source, string values must be enclosed in double quotation marks, for example, `setProxyPassword("secret")`.

```
GoogleAnalytics4DataSource mds = new GoogleAnalytics4DataSource();
mds.setDescription("My Google Analytics 4 Data Source");
mds.setAddTables("'{myTableDefinitionString}'");
mds.setClientID("ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com");
mds.setClientSecret("12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd");
mds.setRefreshToken("1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831");
```

Finally, you will need to configure the example application to print out the data source attributes. Note that this code is specific to the driver and should only be used in the example application. For example, you would add the following section for the minimum properties required to establish a connection:

```
if (ds instanceof GoogleAnalytics4DataSource)
{
GoogleAnalytics4DataSource jmds = (GoogleAnalytics4DataSource) ds;
System.out.println("description=" + jmds.getDescription());
System.out.println("addtables=" + jmds.getAddTables());
System.out.println("clientid=" + jmds.getClientID());
System.out.println("clientsecret=" + jmds.getClientSecret());
System.out.println("refreshtoken=" + jmds.getRefreshToken());
...
System.out.println();
}
```

Calling a data source in an application

Applications can call a Progress DataDirect data source using a logical name to retrieve the `javax.sql.DataSource` object. This object loads the specified driver and can be used to establish a connection to the database.

Once the data source has been registered with JNDI, it can be used by your JDBC application as shown in the following code example.

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("EmployeeDB");
Connection con = ds.getConnection("domino", "spark");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the data source (`EmployeeDB`). The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.DataSource` object. Then, the `DataSource.getConnection()` method is called to establish a connection.

Testing a data source connection

You can use DataDirect Test™ to establish and test a data source connection. The screen shots in this section were taken on a Windows system.

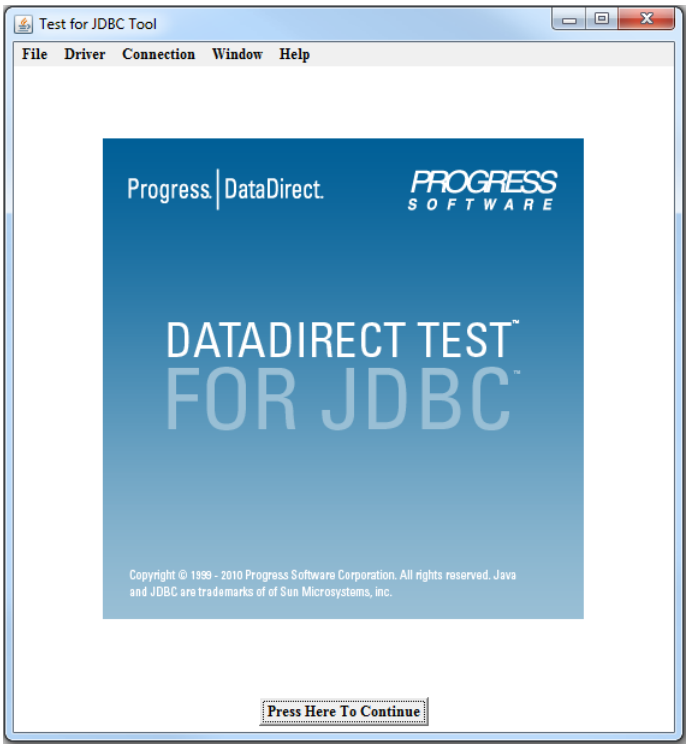
Take the following steps to establish a connection.

1. Navigate to the installation directory. The default location is:
   - Windows systems: `Program Files\Progress\DataDirect\JDBC\testforjdbc`
   - UNIX and Linux systems: `/opt/Progress/DataDirect/JDBC/testforjdbc`

   Note: For UNIX/Linux, if you do not have access to `/opt`, your home directory will be used in its place.

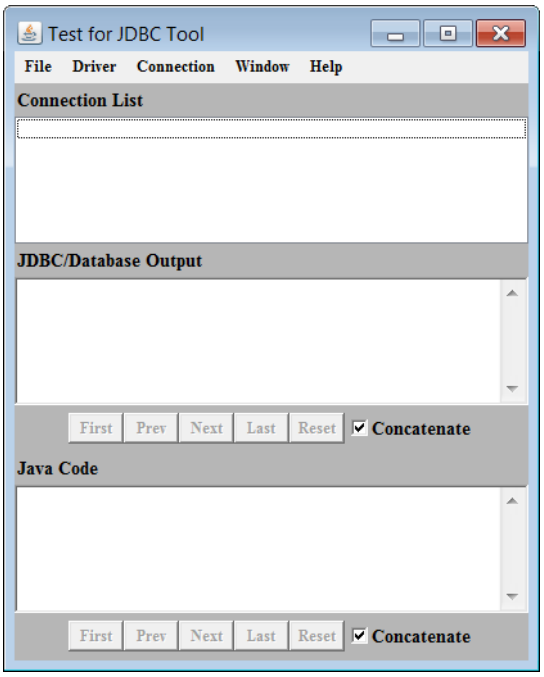
2. From the `testforjdbc` folder, run the platform-specific tool:
   - `testforjdbc.bat` (on Windows systems)
   - `testforjdbc.sh` (on UNIX and Linux systems)

   The **Test for JDBC Tool** window appears:



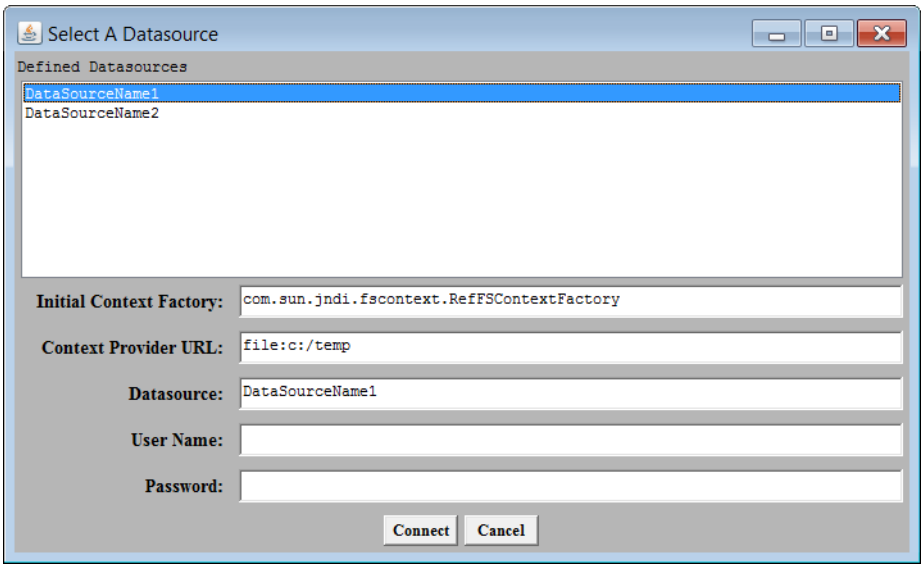
3. Click **Press Here to Continue**.

   The main dialog appears:



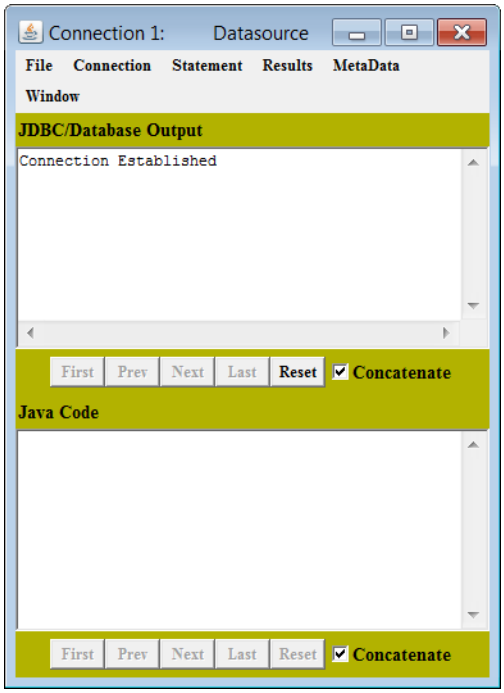
4. From the menu bar, select **Connection > Connect to DB via Data Source**.

   The **Select A Database** dialog appears:



5. Select a datasource template from the **Defined Datasources** field.
6. Provide the following information:
   a) In the **Initial Context Factory**, specify the location of the initial context provider for your application.
   b) In the **Context Provider URL**, specify the location of the context provider for your application.
   c) In the **Datasource** field, specify the name of your datasource.
7. If you are using user ID/password authentication, enter your user ID and password in the corresponding fields.
8. Click **Connect**.

If the connection information is entered correctly, the **JDBC/Database Output** window reports that a connection has been established. If a connection is not established, the window reports an error.



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 properties 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*.

See also

Connection property descriptions on page 51

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 Obtain 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 properties.

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. Open the Configuration Manager by double-clicking the driver jar file. Alternately, from the command line, navigate to the directory containing your driver jar file, and then execute the following command:

   ```
   java -jar googleanalytics4.jar
   ```

   Step result: The Configuration Manager opens in your default web browser.

2. Enter values for the required connection properties under 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`

3. Enter any additional values under the **Connection** tab, or other tabs, as desired.
4. Retrieve access and refresh tokens:
   a. Click **Fetch OAuth Token**.
   b. If prompted, enter your 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.
5. 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 JDBC application to access Google Analytics resources.

Note: Not all the values in the resulting connection string may be required. However, the connection string can be copied directly into your JDBC application. The driver ignores any values that are not required.

Configuring the driver to use OAuth 2.0

Prerequisites:

- Client application registered with Google Cloud
- The client ID and client secret
- 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:

- Set the AddTables property to specify the JSON string that defines custom tables, including dimensions and metrics, that appear in the Google Analytics 4 schema. For example: `'{"TestTable":["sessions","totalUsers","_browser","_sessionSource"]}'`.

  Note: You can generate a value for the property using the **Configure Logical Schema** tool in the Configuration Manager. See Generating connection URLs with the Configuration Manager for details.

- Set either of the following properties:
  - Set the AccessToken property 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 RefreshToken property 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.

  Note: The AccessToken and RefreshToken properties may both be specified. If a value for the AccessToken property is not specified, the driver uses the value of the RefreshToken property 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 AccessToken value and uses the RefreshToken value to generate a new AccessToken value.

  See Obtain access and refresh tokens using the Configuration Manager for details on how to obtain an access or refresh token using the driver's Configuration Manager.

- Set the ClientID property to specify the client ID for your application. See Registering your application with Google Cloud or Obtaining the client ID and client secret.
- Set the ClientSecret property to specify the client secret for your application. See Registering your application with Google Cloud or Obtaining the client ID and client secret.
- (Optionally) Set the Scope property to specify a space-separated list of OAuth scopes that limit the permissions granted by an access token. The default value is:

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

Example URL

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googleanalytics4: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;Scope=scope;");
```

Example data source

```
GoogleAnalytics4DataSource mds = new GoogleAnalytics4DataSource();
mds.setDescription("My Google Analytics 4 Data Source");
mds.setAddTables("'{myTableDefinitionString}'");
mds.setAccessToken("abcDEF123ghi-456Jklmno789-Pqrst01234");
mds.setClientID("ab123c45-def6-7g89-gh1i-m2345no67891.apps.googleusercontent.com");
mds.setClientSecret("12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd");
mds.setRefreshToken("1//12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831");
mds.setScope("scope");
```

Data encryption

All communication between the driver and Google Analytics is encrypted using TLS/SSL.

Important: The driver complies with FIPS when FIPS mode is enabled with the client JVM. See "FIPS (Federal Information Processing Standard)" for more information.

See also

FIPS (Federal Information Processing Standard) on page 49

FIPS (Federal Information Processing Standard)

The Federal Information Processing Standard (or FIPS) is a cryptography standard created by the U.S. government. FIPS specifications require certain secure algorithms, cryptographic modules, and random number generation. The driver is FIPS compliant for data encryption when FIPS is enabled for the JVM on the client machine.

The following applies when the driver is running in a FIPS environment:

- The driver complies with 140-3 and 140-2 standards.
- The driver uses PKCS #11 providers to access keystores.

The driver was tested with FIPS 140-3 enabled using Red Hat OpenJDK 21 on a Red Hat Universal Base Image 9 instance.

Performance considerations

FetchSize: FetchSize can be used to adjust the trade-off between throughput and response time. In general, setting larger values for FetchSize will improve throughput, but can reduce response time. You should set FetchSize to suit your environment. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes improve overall fetch times at the cost of additional memory.

4

Connection property descriptions

You can use connection properties to customize the driver for your environment. This section organizes connection properties according to functionality. You can use connection properties with either the JDBC `DriverManager` or a JDBC data source. For a `DriverManager` connection, a property is expressed as a key value pair and takes the form *`property=value`*. For a data source connection, a property is expressed as a JDBC method and takes the form `set`*`property`*`(`*`value`*`)`.

Note:

- In a JDBC data source, string values must be enclosed in double quotation marks, for example, `setProxyPassword("secret")`.
- The data type listed for each connection property is the Java data type used for the property value in a JDBC data source.
- Connection property names are case-insensitive. For example, `AuthenticationMethod` is the same as `authenticationmethod`.
- For connection properties that support string values, use the following escape sequence to specify values containing leading or trailing spaces and curly brackets: `{`*`value`*`}`. For example: `ProxyPassword={hello }` or `ProxyPassword={{hello}}`.

The following tables describe the connection properties by functionality:

- Additional properties

OAuth 2.0 authentication properties

The following table summarizes the connection properties required to connect to a service when using OAuth 2.0 authentication.

Table 4: Required properties

Property	Data Source Method	Default
AccessToken on page 55	`getAccessToken()` `setAccessToken(String)`	No default value
AuthenticationMethod on page 57	`getAuthenticationMethod()` `setAuthenticationMethod(String)`	`OAuth2`
AuthURI on page 58	`getAuthUri()` `AuthUri(String)`	https://accounts.google.com/o/oauth2/auth
ClientID on page 58	`getClientId()` `setClientId(String)`	No default value
ClientSecret on page 59	`getClientSecret()` `setClientSecret(String)`	No default value
RedirectURI on page 68	`getRedirUri()` `setRedirUri(String)`	No default value
RefreshToken on page 69	`getRefreshToken()` `setRefreshToken(String)`	No default value
Scope on page 69	`getScope()` `setScope(String)`	No default value
ServerName on page 70	`getServerName()` `setServerName(String)`	`analyticsdata.googleapis.com`
TokenURI on page 75	`getTokenUri()` `setTokenUri(String)`	https://accounts.google.com/o/oauth2/token

Mapping properties

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

Property	Data Source Method	Default
AddTables on page 56	`getAddTables()` `setAddTables(String)`	No default value
KeywordConflictSuffix on page 64	`getKeywordConflictSuffix()` `setKeywordConflictSuffix(String)`	No default value

Proxy server properties

The following table summarizes the proxy server connection properties.

Table 5: Proxy Server Properties

Property	Data Source Method	Default
ProxyHost on page 66	`getProxyHost()` `setProxyHost(String)`	No default value
ProxyPassword on page 66	`getProxyPassword()` `setProxyPassword(String)`	No default value
ProxyPort on page 67	`getProxyPort()` `setProxyPort(Integer)`	`0` which means the default is determined by the ProxyHost property. For HTTP URLs: `80` For HTTPS URLs: `443`
ProxyUser on page 67	`getProxyUser()` `setProxyUser(String)`	No default value

Web service properties

The following table summarizes the web service connection properties.

Table 6: Web Service Properties

Property	Data Source Method	Default
StmtCallLimit on page 73	`getStmtCallLimit()` `setStmtCallLimit(Integer)`	`0` (no limit)

Property	Data Source Method	Default
StmtCallLimitBehavior on page 74	`getStmtCallLimitBehavior()` `setStmtCallLimitBehavior(String)`	`errorAlways`
WSRetryCount on page 76	`getWSRetryCount()` `setWSRetryCount(Integer)`	`5`
WSTimeout on page 77	`getWSTimeout()` `setWSTimeout(Integer)`	`120`

Additional properties

The following table summarizes additional connection properties.

Table 7: Additional Properties

Property	Data Source Method	Default
DebugRecord on page 60	`getDebugRecord()` `setDebugRecord(String)`	No default value
DefaultQueryOptions on page 61	`getDefaultQueryOptions()` `setDefaultQueryOptions(String)`	No default value
FetchSize on page 63	`getFetchSize()` `setFetchSize(Integer)`	`100` (rows)
InitializationString on page 64	`getInitializationString()` `setInitializationString(String)`	No default value
LogConfigFile on page 65	`getLogConfigFile()` `setLogConfigFile(String)`	`ddlogging.properties`
SpyAttributes on page 71	`getSpyAttributes()` `setSpyAttributes(String)`	No default value
TransactionMode on page 75	`getTransactionMode()` `setTransactionMode(String)`	`NoTransactions`

For details, see the following topics:

- AccessToken

AccessToken

Purpose

Specifies the access token used to authenticate to Google Analytics 4 with OAuth 2.0 enabled. Typically, this property 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 property 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 expire ten minutes after generation. Once connected, the access token remains valid till the session is disconnected.
- See "OAuth 2.0 authentication" for examples and more information.

Data Source Methods

```
public String getAccessToken()
```

```
public void setAccessToken(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 Authentication on page 43

AddTables

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.

Data Source Methods

```
public String getAddTables()
```

```
public void setAddTables(String)
```

Default Value

No default value

Data Type

String

AuthenticationMethod

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 property does not need to be specified. However, it does appear on the Connection tab of the Configuration Manager.

Valid Values

`OAuth2`

Behavior

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

Data Source Methods

```
public String getAuthenticationMethod()
```

```
public void setAuthenticationMethod(String)
```

Default Value

`OAuth2`

Data Type

String

See also

OAuth 2.0 Authentication on page 43

AuthURI

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.

Data Source Methods

```
public String getAuthUri()
```

```
public void setAuthUri(String)
```

Default Value

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

Data Type

String

See also

OAuth 2.0 Authentication on page 43

ClientID

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.

Data Source Methods

```
public String getClientId()
public void setClientId(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 Authentication on page 43

ClientSecret

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.

Data Source Methods

```
public String getClientSecret()
public void setClientSecret(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 Authentication on page 43

DebugRecord

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`.

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 JDBC Drivers Reference*.
- For assistance, contact Technical Support.

Data Source Methods

```
public String getDebugRecord()
```

```
public void setDebugRecord(String)
```

Default Value

No default value

Data Type

String

DefaultQueryOptions

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

Data Source Methods

```
public String getDefaultQueryOptions()
```

```
public void setDefaultQueryOptions(String)
```

Default Value

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

Data Type

String

ExtendedOptions

Purpose

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

Valid Values

property`=`*value*`[;`*property*`=`*value*`;...]`

where:

property

is a connection property.

value

is the value or setting of the connection property.

Data Source Methods

```
public String getExtendedOptions()
```

```
public void setExtendedOptions(String)
```

Default Value

No default value

Data Type

String

FetchSize

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 FetchSize 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 FetchSize when the result set contains exceptionally wide rows. Alternatively, the driver processes the number of rows specified by FetchSize when the result set contains rows of unexceptional width.
- FetchSize can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.
- You can use FetchSize to reduce demands on memory and decrease the likelihood of out-of-memory errors. Simply, decrease FetchSize to reduce the number of rows the driver is required to process before returning data to the application.

Data Source Methods

```
public Integer getFetchSize()
```

```
public void setFetchSize(Integer)
```

Default Value

`100`

Data Type

Integer

InitializationString

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 property is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified.

Data Source Methods

```
public String getInitializationString()
```

```
public void setInitializationString(String)
```

Default Value

No default value

Data Type

String

KeywordConflictSuffix

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.

Data Source Methods

```
public String getKeywordConflictSuffix()
```

```
public void setKeywordConflictSuffix(String)
```

Default Value

No default value

Data Type

String

LogConfigFile

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 "Using Java Logging" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Methods

```
public String getLogConfigFile()
```

```
public void setLogConfigFile(String)
```

Default Value

ddlogging.properties

Data Type

String

ProxyHost

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.

Data Source Methods

```
public String getProxyHost()
public void setProxyHost(String)
```

Default Value

No default value

Data Type

String

ProxyPassword

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.

Data Source Methods

```
public String getProxyPassword()
public void setProxyPassword(String)
```

Default Value

No default value

Data Type

String

ProxyPort

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.

Data Source Methods

```
public Integer getProxyPort()
```

```
public void setProxyPort(Integer)
```

Default Value

`0` which means that the default value is determined by whether the value specified for the ProxyHost property is an HTTP or HTTPS URL.

For HTTP: `80`

For HTTPS: `443`

Data Type

Integer

ProxyUser

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.

Data Source Methods

```
public String getProxyUser()
public void setProxyUser(String)
```

Default Value

No default value

Data Type

String

RedirectURI

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.

Data Source Methods

```
public String getRedirUri()
public void setRedirUri(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 Authentication on page 43

RefreshToken

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 Google Analytics 4 instance.

Notes

- See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getRefreshToken()
```

```
public void setRefreshToken(String)
```

Default Value

No default value

Data Type

String

Scope

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.

Data Source Methods

```
public String getScope()
```

```
public void setScope(String)
```

Default Value

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

Data Type

String

ServerName

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 HostName property is an alias for the Server Name (ServerName) property.

Data Source Methods

```
public String getServerName()
```

```
public void setServerName(String)
```

Default Value

```
https://analyticsdata.googleapis.com
```

Data Type

String

SpyAttributes

Purpose

Enables DataDirect Spy to log detailed information about calls that are issued by the driver on behalf of the application. DataDirect Spy is not enabled by default.

Valid Values

(*spy_attribute*[;*spy_attribute*]...)

where:

spy_attribute

is any valid DataDirect spy attribute.

Behavior

Attribute	Description
linelimit=*numberofchars*	Sets the maximum number of characters that DataDirect Spy logs on a single line. The default is 0 (no maximum limit).
log=(file)*filename*	Directs logging to the file specified by *filename*. For Windows, if coding a path to the log file in a Java string, the backslash character (\) must be preceded by the Java escape character, a backslash. For example: log=(file)C:\\temp\\spy.log;logIS=yes;logTName=yes.

Attribute	Description
log=(filePrefix)*file_prefix*	Directs logging to a file prefixed by *file_prefix*. The log file is named *file_prefixX*.log where: *X* is an integer that increments by 1 for each connection on which the prefix is specified. For example, if the attribute log=(filePrefix)C:\\temp\\spy_ is specified on multiple connections, the following logs are created: C:\temp\spy_1.log C:\temp\spy_2.log C:\temp\spy_3.log ... If coding a path to the log file in a Java string, the backslash character (\) must be preceded by the Java escape character, a backslash. For example: log=(filePrefix)C:\\temp\\spy_;logIS=yes;logTName=yes.
log=System.out	Directs logging to the Java output standard, System.out.
logIS={ yes | no | nosingleread }	Specifies whether DataDirect Spy logs activity on InputStream and Reader objects. When logIS=nosingleread, logging on InputStream and Reader objects is active; however, logging of the single-byte read InputStream.read or single-character Reader.read is suppressed to prevent generating large log files that contain single-byte or single character read messages. The default is no.
logLobs={ yes | no }	Specifies whether DataDirect Spy logs activity on BLOB and CLOB objects.
logTName={ yes | no }	Specifies whether DataDirect Spy logs the name of the current thread. The default is no.
timestamp={ yes | no }	Specifies whether a timestamp is included on each line of the DataDirect Spy log. The default is no.

Example

The following value instructs the driver to log all JDBC activity to a file using a maximum of 80 characters for each line.

```
(log=(file)/tmp/spy.log;linelimit=80)
```

Notes

- If coding a path on Windows to the log file in a Java string, the backslash character (\) must be preceded by the Java escape character, a backslash. For example: `log=(file)C:\\temp\\spy.log`.
- If a log file name does not include the `.log` extension, the driver automatically appends it. For example, a file named `spy.jsp` is renamed to `spy.jsp.log` by the driver.
- For more information, refer to "Tracking JDBC calls with DataDirect Spy" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Methods

```
public String getSpyAttributes()
```

```
public void setSpyAttributes(String)
```

Default Value

No default value

Data Type

String

StmtCallLimit

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 StmtCallLimitBehavior property.

Data Source Methods

```
public Integer getStmtCallLimit()
```

```
public void setStmtCallLimit(Integer)
```

Default Value

0

Data Type

Integer

See also

StmtCallLimitBehavior on page 74

StmtCallLimitBehavior

Purpose

Specifies the behavior of the driver when the maximum web service call limit specified by the StmtCallLimit property is exceeded.

Valid Values

ReturnResults | ErrorAlways

Behavior

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

Data Source Methods

```
public String getStmtCallLimitBehavior()
```

```
public void setStmtCallLimitBehavior(String)
```

Default Value

ErrorAlways

Data Type

String

See also

StmtCallLimit on page 73

TokenURI

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.

Data Source Methods

```
public String getTokenUri()
public void setTokenUri(String)
```

Default Value

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

Data Type

String

See also

OAuth 2.0 Authentication on page 43

TransactionMode

Purpose

Specifies how the driver handles manual transactions.

Valid Values

`NoTransactions` | `Ignore`

Behavior

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

If set to `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.

Data Source Methods

```
public String getTransactionMode()
```

```
public void setTransactionMode(String)
```

Default Value

`NoTransactions`

Data Type

String

WSRetryCount

Purpose

Specifies the number of times the driver retries a timed-out Select request. The timeout period is specified by the WSTimeout connection property.

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.

Data Source Methods

```
public Integer getWSRetryCount()
```

```
public void setWSRetryCount(Integer)
```

Default Value

`5`

Data Type

Integer

See also

WSTimeout on page 77

WSTimeout

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 WSRetryCount is set to retry timed-out requests, the driver retries the request the specified number of times.

Data Source Methods

```
public Integer getWSTimeout()
```

```
public void setWSTimeout(Integer)
```

Default Value

`120`

Data Type

Integer

5

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 8: 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()`

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 ...}
```

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 82
From clause on page 84
Where clause on page 86
Group By clause on page 87
Having clause on page 87
Union operator on page 88
Intersect operator on page 88
Except and Minus operators on page 89
Order By clause on page 90
Limit clause on page 91

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 94
Aggregate functions on page 83

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 9: 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 86

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.

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

Subqueries on page 91

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
```

See also

Subqueries on page 91
SQL expressions on page 94

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 91
SQL expressions on page 94

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
```

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 91
SQL expressions on page 94

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 10: 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`
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 11: Arithmetic Operators

Operator	Purpose	Example
+ -	Denotes a positive or negative expression. These are unary operators.	`SELECT * FROM emp WHERE comm = -1`
* /	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 12: 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 13: 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`
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`

Operator	Purpose	Example
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 14: 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 15: 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, including String, Numeric, Timedate, and System functions.

Refer to "Scalar functions" in the *Progress DataDirect for JDBC 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 16: 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`
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`

6

Introduction to the Google Analytics 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

To query your data, you must add tables to your Google Analytics relational schema. Custom tables can be added to your Google Analytics relational schema, and further edited, 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 AddTables property. 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 `rowId`, `propertyID`, `startDate`, `endDate`, `currencyCode`, 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 DefaultQueryOptions property.

- `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	