Progress DataDirect for JDBC for Google BigQuery

User's Guide

Release 6.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for JDBC for Google BigQuery Driver

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

The documentation for the driver also includes the *Progress DataDirect for 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-bigquery.

For details, see the following topics:

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

- **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 45 for details.
  - The driver has been enhanced to retry API call executions when an HTTP failure or driver exception occurs. You can configure this behavior with the new RetryExceptions connection property. When this property is set to `1`, the driver uses the retry value specified by the WSRetryCount connection property. See RetryExceptions on page 90 for details.
  - The driver has been enhanced to support fetching access and refresh tokens at connection when OAuth2.0 is enabled. When using the new dynamic authorization code grant, you can initiate an authorization code grant flow by specifying login credentials using the login prompt for your service, thereby providing a method to authenticate without fetching access and refresh tokens via the Configuration Manager or a third-party application. In addition, the new EnableLoginPrompt and RedirectURI properties have been added to enable this functionality. See Dynamic authorization code grant on page 40, EnableLoginPrompt on page 70, RedirectURI on page 87 and for details.
  - The driver has been enhanced to support the JSON and Interval data types.
  - The driver has been enhanced to allow users to specify values for the following connection properties:
    - AuthURI on page 58 and TokenURI on page 102 for OAuth 2.0 authentication
    - JWTAudience on page 75 and TokenURI on page 102 for Service Account authentication

    Earlier, these properties were hidden and their default values were used for authenticating to Google BigQuery. See Configuring OAuth 2.0 authentication on page 39 and Configuring service account authentication on page 41 for details.
  - The driver has been enhanced to support the Google BigQuery Streaming API when executing batch inserts. This behavior can be configured using the UseStreamingInsert connection property. See Google BigQuery Streaming API for details.

- The driver has been enhanced with the new PrimaryKeyPattern connection property, which allows you to determine which column in a table is designated as the primary key. Google BigQuery does not have the concept of primary keys, or even uniqueness. However, some applications do not function properly without at least one column in a table designated as the primary key. This property allows your applications that require a primary key to function correctly when connecting to Google BigQuery data sources.
- The driver has been enhanced to support the BIGNUMERIC data type. See Data types on page 16 for more information.
- The ServiceAccountKeyContent connection property has been added to the driver. This property allows you to specify the private key required for service account authentication without having to persist the `.json` or `.p12` file that contains the private key. See Configuring service account authentication on page 41 and ServiceAccountKeyContent on page 95 for details.
- The EnableCatalogSupport connection property has been added to the driver. It determines whether the driver supports specifying values for catalog parameters in metadata calls.
- The driver has been enhanced to support the Google BigQuery Storage API when fetching large result sets. See Google BigQuery Storage API and Storage API properties for details.

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

- The driver has been enhanced to include timestamp in the Spy and JDBC packet logs by default. If required, you can disable the timestamp logging by specifying the following at connection. For Spy logs, set `spyAttributes=(log=(file)Spy.log;timestamp=no)` and for JDBC packet logs, set `ddtdbg.ProtocolTraceShowTime=false`.
- Interactive SQL for JDBC (JDBCISQL) is now installed with the product. JDBCISQL is a command-line interface that supports connecting your driver to a data source, executing SQL statements and retrieving results in a terminal. This tool provides a method to quickly test your drivers in an environment that does not support GUIs.
- **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 97 for details.
  - The default value of the WSRetryCount connection option has been changed to `5`.
  - The default value of the SchemaSet configuration option has been changed to the project and dataset specified at connection. For details, see SchemaSet (Configuration Option) on page 64

Highlights of 6.0.0 Release

- The driver supports both standard and legacy Google BigQuery SQL dialects. See Standard and legacy SQL support on page 15 for more information.
- The driver supports JDBC core functions. For details, refer to "JDBC support" in the *Progress DataDirect for JDBC Drivers Reference*.
- The driver serves as a complete pass-through driver. It leverages Google BigQuery SQL engine to execute queries.
- The driver supports create, read, update, and delete (CRUD) operations.

- The driver returns data for complex data types, such as Array and Struct, as JSON strings, which are easy to comprehend for the JDBC applications. See JsonFormat on page 74 for more information.
- The driver provides proxy support. See ProxyHost on page 84, ProxyPassword on page 85, ProxyPort on page 86, and ProxyUser on page 86 for more information.
- The driver supports the handling of large result sets with configurable paging and the FetchSize on page 71 and WSFetchSize on page 104 connection properties.

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\googlebigquery.jar
     ```
   - **UNIX/LINUX Example**

     ```
     CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/googlebigquery.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. For example:

```
jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy;
```

Note: See Authentication on page 39 to know more about user ID/password authentication and other authentication methods you can use to connect to the server.

- **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 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 the processes for testing `DriverManager` and data source connections, and the SQL statements supported by the driver.
   - Testing a DriverManager connection: This section discusses how to establish and test a `DriverManager` connection using DataDirect Test.
   - Testing a data source connection: This section discusses how to establish and test a data source connection using DataDirect Test.

This completes the deployment of the driver.

Driver and DataSource classes

The `Driver` class for the driver is:

`com.ddtek.jdbc.googlebigquery.GoogleBigQueryDriver`

The `DataSource` class for the driver is:

`com.ddtek.jdbcx.googlebigquery.GoogleBigQueryDataSource`

Connection URL

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

```
jdbc:datadirect:googlebigquery:Project=project;Dataset=dataset;
AccessToken=accesstoken;RefreshToken=refreshtoken;ClientID=clientid;
ClientSecret=clientsecret[;property=value[;...]]
```

Note:

- Connection property names are case-insensitive. For example, `Project` is the same as `project`.
- 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: `Project={myproject }` or `Project={{myproject}}`.

---

where:

`Project`

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

`Dataset`

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

`AccessToken`

specifies the access token required to authenticate to a Google BigQuery instance.

`RefreshToken`

specifies the refresh token used to either request a new access token or renew an expired access token.

`ClientID`

specifies the consumer key for your application.

`ClientSecret`

specifies the consumer secret for your application.

The following example shows how to establish a connection to a Google BigQuery instance.

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy");
```

See also

Connection property descriptions on page 51

Connection properties

The driver includes over 40 connection properties. You can use connection properties to customize the driver for your environment. Connection properties can be used to accomplish different tasks, such as implementing driver functionality and optimizing performance. You can specify connection properties in a connection URL or within a JDBC data source object.

See Connection property descriptions on page 51 for more information.

See also

Connection property descriptions on page 51

Standard and legacy SQL support

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

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

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

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

See also

Syntax on page 101

Using identifiers

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

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

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

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

Data types

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

Table 1: Google BigQuery Data Types

Google BigQuery Data Type	JDBC Data Type
ARRAY	VARCHAR
BIGNUMERIC	DECIMAL
BOOL	BOOLEAN
BYTES	VARBINARY
DATE	DATE
DATETIME	TIMESTAMP
FLOAT64	DOUBLE
GEOGRAPHY	VARCHAR
INTERVAL	VARCHAR
INT64	BIGINT
JSON	LONGVARCHAR
NUMERIC	DECIMAL
RECORD	VARCHAR
STRING	VARCHAR
TIME	TIME
TIMESTAMP	TIMESTAMP

getTypeInfo()

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

Table 2: getTypeInfo() Results

TYPE_NAME = ARRAY AUTO_INCREMENT = false CASE_SENSITIVE = true CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = ARRAY MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = BIGNUMERIC 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 = BIGNUMERIC MAXIMUM_SCALE = 77	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 77 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = false
TYPE_NAME = BOOL 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 = BOOL MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = BYTES AUTO_INCREMENT = false CASE_SENSITIVE = false CREATE_PARAMS = NULL DATA_TYPE = -3 (VARBINARY) FIXED_PREC_SCALE = false LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = BYTES MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 65535 SEARCHABLE = 3 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 = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 10 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = DATETIME AUTO_INCREMENT = false CASE_SENSITIVE = false CREATE_PARAMS = NULL DATA_TYPE = 93 (TIMESTAMP) FIXED_PREC_SCALE = false LITERAL_PREFIX = DATETIME' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = DATETIME MAXIMUM_SCALE = NULL	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 26 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = FLOAT64 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 = FLOAT64 MAXIMUM_SCALE = NULL	MINIMUM_SCALE = -324 NULLABLE = 1 NUM_PREC_RADIX = 2 PRECISION = 15 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = false
TYPE_NAME = GEOGRAPHY AUTO_INCREMENT = false CASE_SENSITIVE = false CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = GEOGRAPHY MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 65535 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = INTERVAL AUTO_INCREMENT = false CASE_SENSITIVE = true CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = INTERVAL MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 128 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = INT64 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 = INT64 MAXIMUM_SCALE = NULL	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 19 SEARCHABLE = 3 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 = -1 (LONGVARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = JSON ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = JSON MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16777216 SEARCHABLE = 0 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = NUMERIC 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 = NUMERIC MAXIMUM_SCALE = 32767	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 38 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = false

TYPE_NAME = RECORD AUTO_INCREMENT = false CASE_SENSITIVE = true CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = RECORD MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 65535 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = STRING AUTO_INCREMENT = false CASE_SENSITIVE = true CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = false LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = STRING MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 65535 SEARCHABLE = 3 SQL_DATA_TYPE = NULL SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

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 = NULL	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 15 SEARCHABLE = 3 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 = DATETIME' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = TIMESTAMP MAXIMUM_SCALE = NULL	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 26 SEARCHABLE = 3 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

Using the driver

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.

For details, see the following topics:

- Required permissions for Java SE with the standard Security Manager enabled
- Connecting from an application
- Connecting through a proxy server
- Performance considerations
- Authentication
- Data encryption
- Internal and external tables support
- Google BigQuery Storage API
- Google BigQuery Streaming API
- Failover support
- Timeouts
- Isolation levels
- Scrollable cursors
- Unicode support

- Error handling
- Parameter metadata support
- ResultSet metadata support

Required permissions for Java SE with the standard Security Manager enabled

Using the driver on a Java platform with the standard Security Manager enabled requires certain permissions to be set in the Java SE security policy file `java.policy`. The default location of this file is *`java_install_dir`*`/jre/lib/security`.

Note: Security manager may be enabled by default in certain scenarios, such as running on an application server or in a Web browser applet.

To run an application on a Java platform with the standard Security Manager, use the following command:

```
"java -Djava.security.manager application_class_name"
```

where *`application_class_name`* is the class name of the application.

Refer to your Java documentation for more information about setting permissions in the security policy file.

Permissions for establishing connections

To establish a connection to the database server, the driver must be granted the permissions as shown in the following example:

```
grant codeBase "file:/install_dir/lib/60/-" {
   permission java.net.SocketPermission "*", "connect";
};
```

where:

`install_dir`

is the product installation directory.

Granting access to Java properties

To allow the driver to read the value of various Java properties to perform certain operations, permissions must be granted as shown in the following example:

```
grant codeBase "file:/install_dir/lib/60/-" {
   permission java.util.PropertyPermission "*", "read, write";
};
```

where:

install_dir

is the product installation directory.

Granting access to temporary files

Access to the temporary directory specified by the JVM configuration must be granted in the Java SE security policy file to use insensitive scrollable cursors or to perform client-side sorting of DatabaseMetaData result sets. The following example shows permissions that have been granted for the `C:\TEMP` directory:

```
grant codeBase "file:/install_dir/lib/60/-" {
// Permission to create and delete temporary files.
// Adjust the temporary directory for your environment.
   permission java.io.FilePermission "C:\\TEMP\\-", "read,write,delete";
};
```

where:

install_dir

is the product installation directory.

Connecting from an application

After the driver has been installed and defined on your class path, 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).

Setting the classpath

Before you can connect, the driver must be defined in your CLASSPATH variable. 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 `googlebigquery.jar` file as shown, where *install_dir* is the path to your product installation directory:

```
install_dir/lib/60/googlebigquery.jar
```

Windows Example

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

UNIX Example

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

Connecting using the JDBC Driver Manager

One way to connect to a Google BigQuery instance is through the JDBC Driver Manager using the `DriverManager.getConnection()` method. This method specifies a string containing a connection URL. This example shows how to establish a connection to a data source:

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy");
```

Passing the connection URL

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

Connection URL Syntax

```
jdbc:datadirect:googlebigquery:Project=project;Dataset=dataset;
AccessToken=accesstoken;RefreshToken=refreshtoken;ClientID=clientid;
ClientSecret=clientsecret[;property=value[;...]]
```

where:

`Project`

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

`Dataset`

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

`AccessToken`

specifies the access token required to authenticate to a Google BigQuery instance.

`RefreshToken`

specifies the refresh token used to either request a new access token or renew an expired access token.

`ClientID`

specifies the consumer key for your application.

`ClientSecret`

specifies the consumer secret for your application.

Connection URL Example

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
```

```
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy");
```

See also

Connection property descriptions on page 51

Testing the connection

You can use DataDirect Test™ to verify your connection. The screen shots in this section were taken on a Windows system.

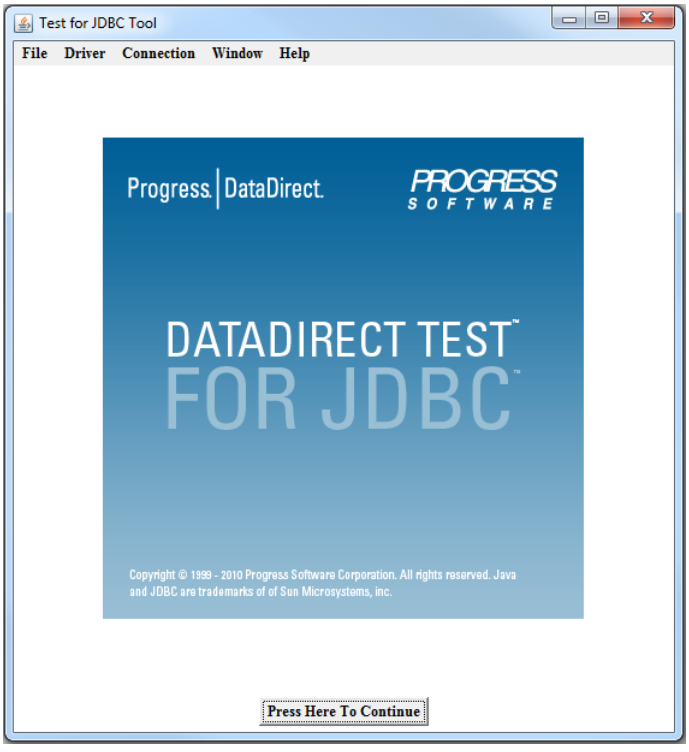
To test the connection from the driver to your data source, follow these steps:

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.

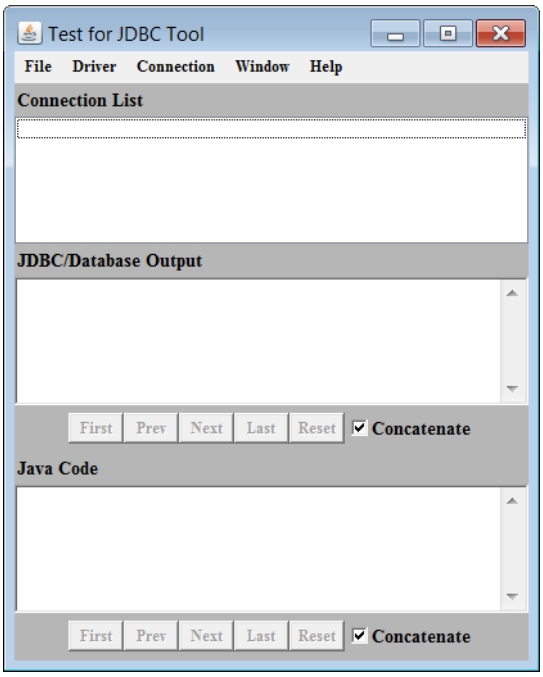
2. From the `testforjdbc` folder, run the platform-specific tool:
   - `testforjdbc.bat` (on Windows systems)
   - `testforjdbc.sh` (on UNIX and Linux systems)

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



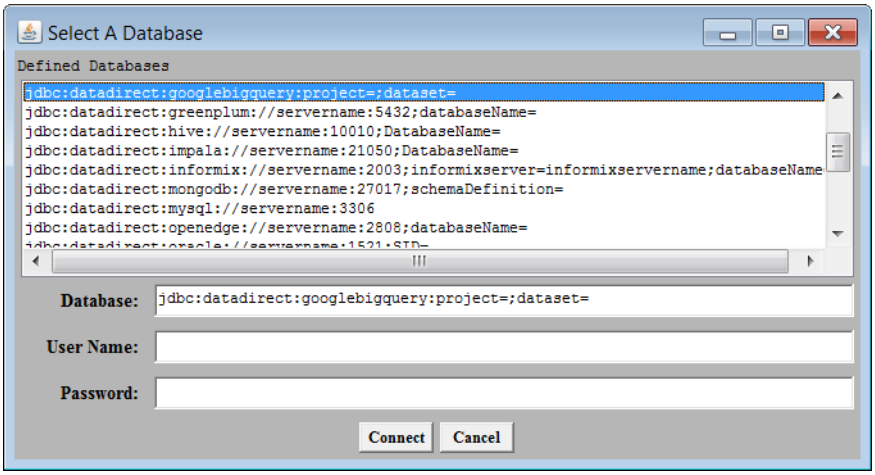
3. Click **Press Here to Continue**.

   The main dialog appears:



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

   The **Select A Database** dialog appears:



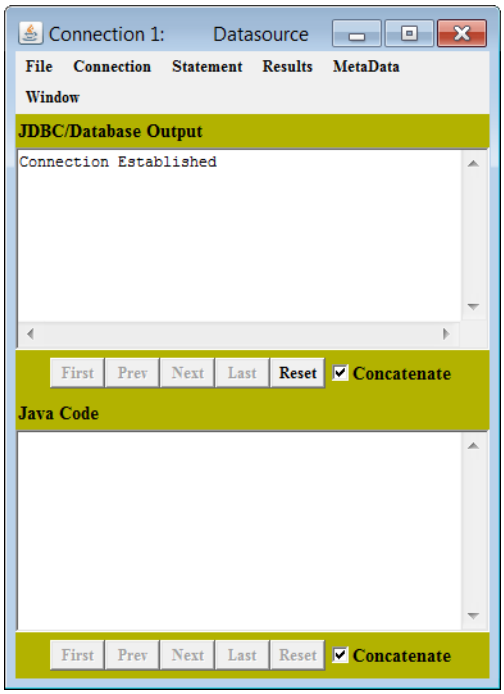
5. Select the appropriate database template from the **Defined Databases** field.
6. In the **Database** field, specify all required connection properties.

   For example:

   ```
   jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
   AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
   ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy
   ```

7. 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.)



Refer to "DataDirect Test" in the *Progress DataDirect for JDBC Drivers Reference* for more information about using DataDirect Test.

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 data source 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.

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 the data source class for the driver:

```
import com.ddtek.jdbcx.googlebigquery.GoogleBigQueryDataSource;
```

Next, you will need to set the values and define the data source. For example, the following definition contains the minimum properties required to establish connection:

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, setProject("myproject").

```
GoogleBigQueryDataSource mds = new GoogleBigQueryDataSource();
mds.setDescription("My Google BigQuery Datasource");
mds.setProject("myproject");
mds.setDataset("mydataset");
mds.setAccessToken("abcdefghi12345678");
mds.setRefreshToken("wxyz123456789");
mds.setClientID("123abc.apps.googleusercontent.com");
mds.setClientSecret("ab123xy");
```

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 GoogleBigQueryDataSource)
{
GoogleBigQueryDataSource jmds = (GoogleBigQueryDataSource) ds;
System.out.println("description=" + jmds.getDescription());
System.out.println("project=" + jmds.getProject());
System.out.println("dataset=" + jmds.getDataset());
System.out.println("accessToken=" + jmds.getAccessToken());
System.out.println("refreshToken=" + jmds.getRefreshToken());
```

```
System.out.println("clientID=" + jmds.getClientID());
System.out.println("clientSecret=" + jmds.getClientSecret());
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();
```

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.

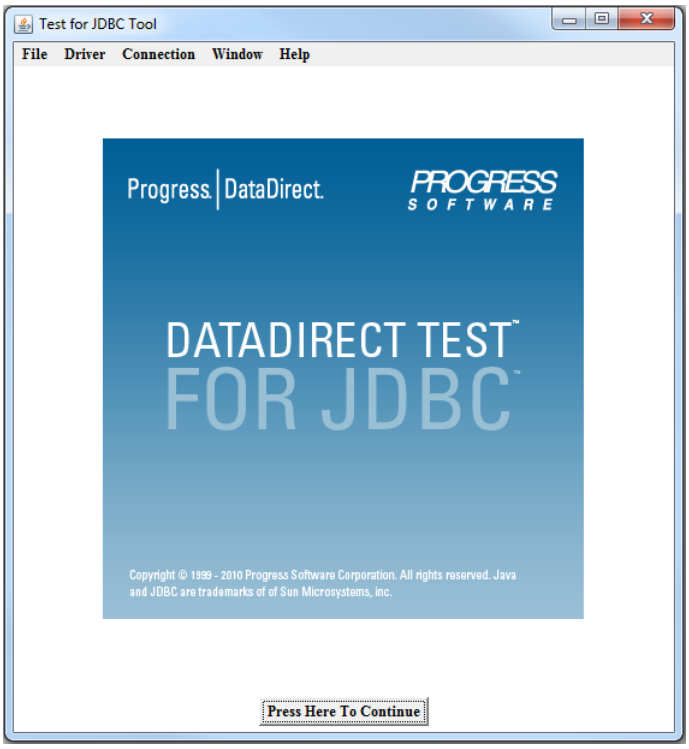
Take the following steps to establish a connection.

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.

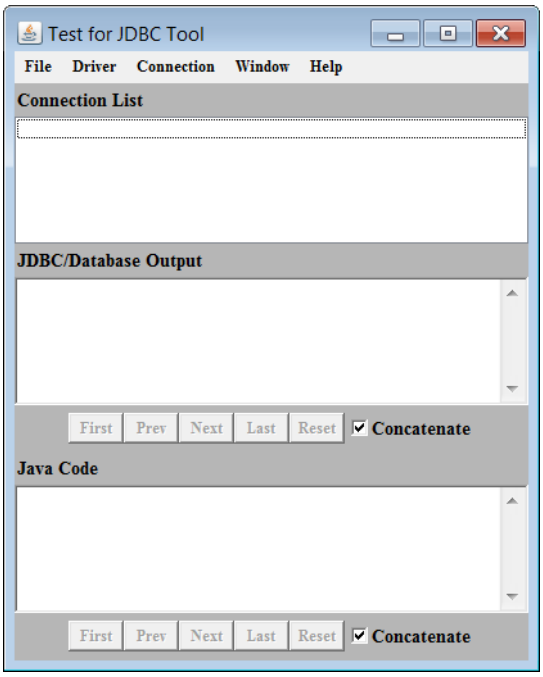
2. From the `testforjdbc` folder, run the platform-specific tool:
   - `testforjdbc.bat` (on Windows systems)
   - `testforjdbc.sh` (on UNIX and Linux systems)

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



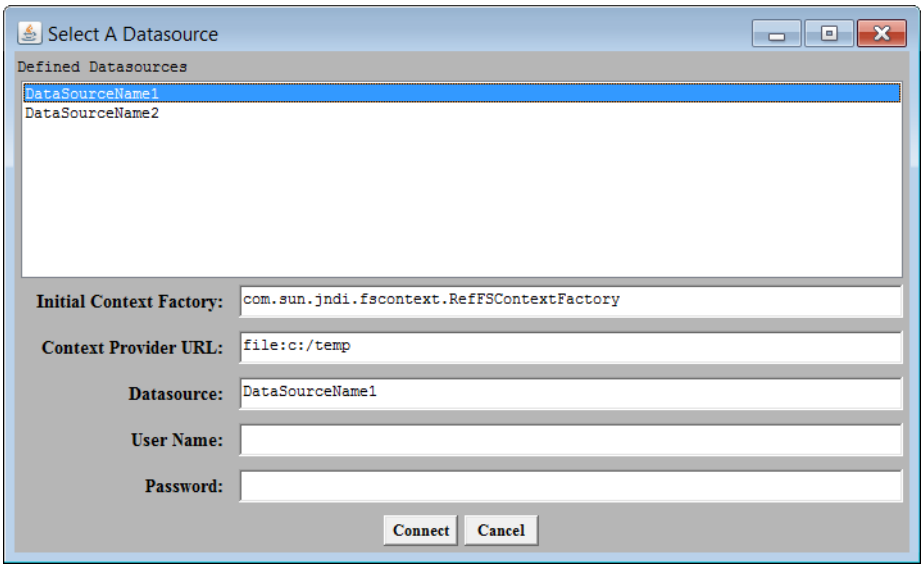
3. Click **Press Here to Continue**.

   The main dialog appears:



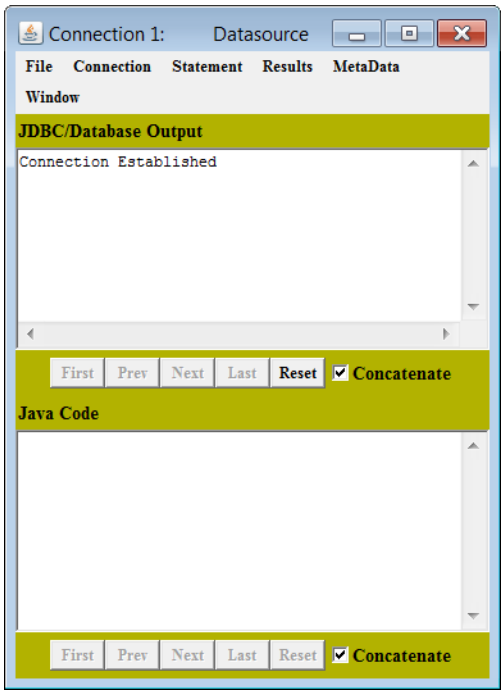
4. From the menu bar, select **Connection > Connect to DB via Data Source**.

   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. 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.



Connecting through a proxy server

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

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

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

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

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

Alternatively, you can use the ProxyHost, ProxyPort, ProxyUser, and ProxyPassword connection properties. See "Connection Property Descriptions" for details about these properties.

See also

Connection property descriptions on page 51

Performance considerations

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

UseStorageAPI: The UseStorageAPI connection property can be used to improve performance by enabling the driver to use the Google BigQuery Storage API when fetching large result sets. When UseStorageAPI is set to `true`, the driver uses the Storage API for selects when the number of rows in the result set exceeds the value of the StorageAPIThreshold property, and the number of pages in the result set exceeds the value of the StorageAPIMinPageCount property.

UseStreamingInsert: The UseStreamingInsert connection property can be used to improve performance by enabling the driver to use the Google BigQuery Streaming API for executing batch inserts. When UseStreamingInserts is set to `true`, the driver uses the Streaming API for insert operations with the `tabledata.insertAll` method.

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

Note: The number specified for WSPoolSize should not exceed the amount of sessions permitted by your Google BigQuery account.

Authentication

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

The Google BigQuery driver supports the following methods of authentication:

- *OAuth 2.0 authentication*. It allows the users to authenticate to a Google BigQuery instance without specifying user ID and password. The driver supports the following OAuth 2.0 flow and grant types for accessing Google BigQuery instances.
  - Access token flow
  - Refresh token grant
  - Dynamic authorization code grant
- *Service account authentication*. It allows the users to authenticate to a Google BigQuery instance using a service account, which is a type of Google account that represents an application instead of an individual end user.

See also

Configuring OAuth 2.0 authentication on page 39
Configuring service account authentication on page 41
Connection property descriptions on page 51

Configuring OAuth 2.0 authentication

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

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

- Set the AuthenticationMethod property to `oauth2`.
- Set at least one of the following properties:
  - AccessToken: Set this to specify the access token you have obtained to authenticate to Google BigQuery.
  - RefreshToken: Set this to specify the refresh token you have obtained to authenticate to Google BigQuery.

  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 uses the AccessToken value; however, if the AccessToken value expires, it uses the RefreshToken value to generate a new AccessToken value.
- Set the ClientID property to specify the consumer key for your application.
- Set the ClientSecret property to specify the consumer secret for your application.
- Optionally, set the AuthURI property to specify the endpoint for obtaining an authorization code from a private endpoint. The default value is `https://accounts.google.com/o/oauth2/auth`.

- Optionally, set the TokenURI property to specify the endpoint for retrieving access tokens. The default value is `https://accounts.google.com/o/oauth2/token`.
- Optionally, set the Scope property to specify the OAuth scope. It limits the permissions granted by an access token.

The following example shows how to connect to a Google BigQuery instance using OAuth 2.0 authentication.

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:AuthenticationMethod=oauth2;
Project=myproject;Dataset=mydataset;AccessToken=abcdefghi12345678;
RefreshToken=wxyz123456789;ClientID=123abc.apps.googleusercontent.com;
ClientSecret=ab123xy");
```

See also

AccessToken on page 59
RefreshToken on page 89
ClientID on page 62
ClientSecret on page 62
AuthURI on page 58
TokenURI on page 102
Scope on page 93
Dynamic authorization code grant on page 40

Dynamic authorization code grant

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

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

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

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

To use dynamic authorization code grant, set the following connection properties:

- Set the AuthenticationMethod property to `oauth2`.
- Set the EnableLoginPrompt property to `true` (enabled). When EnableLoginPrompt is enabled, the driver launches the login prompt for your service in a separate browser window to initiate the OAuth grant flow.
- Set the ClientID property to specify the client ID key for your application.
- Set the ClientSecret property to specify the client secret for your application.

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

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

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

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:AuthenticationMethod=oauth2;
EnableLoginPrompt=true;ClientID=123abc.apps.googleusercontent.com;
ClientSecret=ab123xy;AuthURI=https://accounts.google.com/o/oauth2/auth;
TokenURI=https://accounts.google.com/o/oauth2/token;
Scope=https://www.googleapis.com/auth/bigquery;
RedirectURI=http://localhost:80;");
```

See also

Connection property descriptions on page 51

Configuring service account authentication

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

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

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

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

- Set the AuthenticationMethod property to `serviceaccount`.
- Set the ServiceAccountEmail property to specify your service account's email address.
- Set either the ServiceAccountKeyContent property or the ServiceAccountPrivateKey property
  - ServiceAccountKeyContent specifies the private key required to authenticate to Google BigQuery. Use this property if you do not want to persist the private key file in your environment.
  - ServiceAccountPrivateKey specifies the full path to the `.json` or `.p12` file that contains the private key. The driver extracts the private key value from the specified file and uses it to authenticate the user to the database. Use this property if it is preferable to persist the private key file.
- Optionally, set the JWTAudience property to specify the JWT audience claim associated with your service account. The default value is `https://accounts.google.com/o/oauth2/token`.
- Optionally, set the TokenURI property to specify the endpoint for retrieving access tokens. The default value is `https://accounts.google.com/o/oauth2/token`.

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

ServiceAccountKeyContent

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:AuthenticationMethod=serviceaccount;
Project=myproject;Dataset=mydataset;ServiceAccountEmail=abc123@iam.gserviceaccount.com;
ServiceAccountKeyContent=NJJXZexIHJFGYBgkqhkiG9w0BAQnWRwiHANpf3MC1pVRqhtTE5tSpxZeQnICG
4zp087Eidn4qc66udg8KAHknyqFdj7b\n+MgxMFPAvJ59cylHFaHA4pGmeGfVqzYub6LEs9aN/751jmZqcuAYp
5nXRF1EvJPN\nsDuJGLvuuDBZW0iux0liEHmcQVBBKwIx8t+EQxePGTiLsBoCdzOUsi4UWWv\nASqfdP/kSX+N;);
```

ServiceAccountPrivateKey

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:googlebigquery:AuthenticationMethod=serviceaccount;
Project=myproject;Dataset=mydataset;ServiceAccountEmail=abc123@iam.gserviceaccount.com;
ServiceAccountPrivateKey=abc123.json")
```

See also

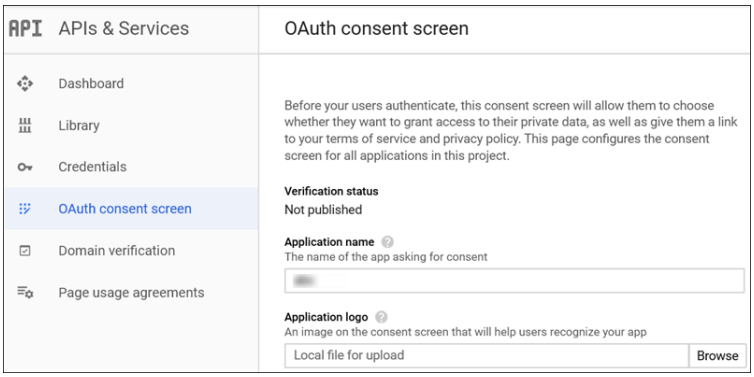
AuthenticationMethod on page 60
ServiceAccountEmail on page 94
ServiceAccountKeyContent on page 95
ServiceAccountPrivateKey on page 96
JWTAudience on page 75
Connection property descriptions on page 51

Generating access tokens and refresh tokens

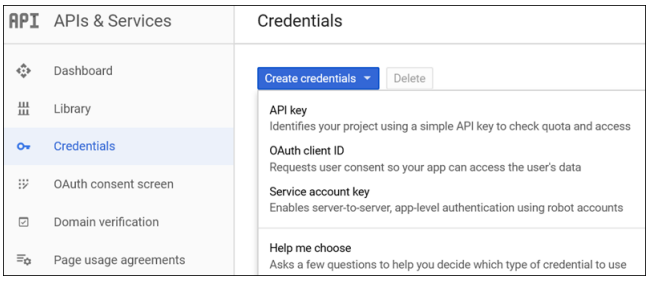
This section guides you through the process of generating access tokens and refresh tokens that are used to authenticate to Google BigQuery when OAuth 2.0 is set as the authentication method (`AuthenticationMethod=oauth2`).

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

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



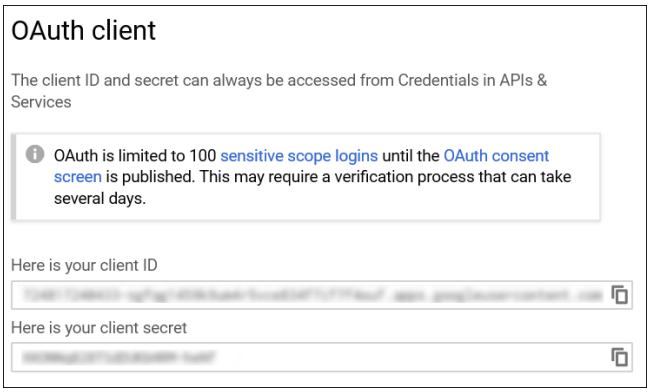
4. On the OAuth consent screen, enter the following details; then, click **Save**:
   - Application name
   - Support email
   - Scopes for Google APIs
   - Authorized domains
   - Application Homepage link
   - Application Privacy Policy link
5. On the left pane, click **Credentials**. Next, from the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



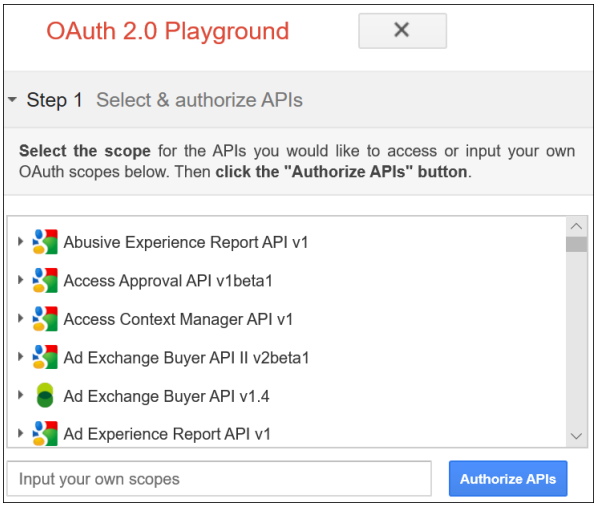
6. The Create OAuth client ID screen appears. Select **Other** as your Application type, enter your application's name, and then click **Create**.

.

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



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



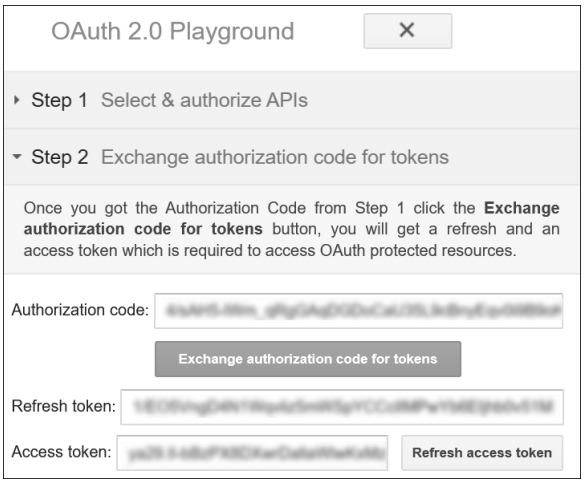
   Login screen appears.

10. Click your username. The following message appears: "Google OAuth 2.0 Playground wants to access your Google Account."
11. Scroll down and click **Allow**.

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

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

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



See also

AccessToken on page 59
RefreshToken on page 89

Data encryption

All communication between the driver and Google BigQuery, including OAuth 2.0 authentication, 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.

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.

Internal and external tables support

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

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

Google BigQuery Storage API

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

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

You can enable the driver to use the Storage API with the UseStorageAPI connection property. You can further configure how the driver uses the Storage API with the StorageAPIThreshold and StorageAPIMinPageCount connection properties.

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

Important: The BigQuery Storage API has a separate pricing model. Refer to "BigQuery Storage pricing" for details: https://cloud.google.com/bigquery/pricing#storage-api.

See also

Storage API Properties on page 53
UseStorageAPI on page 103
StorageAPIMinPageCount on page 100
StorageAPIThreshold on page 100

Google BigQuery Streaming API

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

You can enable the driver to use the Streaming API with the `UseStreamingInsert` connection property. When this connection property is set to `true`, the driver uses the `tabledata.insertAll` method for streaming batched inserts. The request payload sent to the API is in a JSON format.

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

Before enabling the driver to use the Streaming API, prerequisites described in the Google Cloud BigQuery Streaming API reference must be met.

See also

UseStreamingInsert on page 104

Failover support

The driver provides connection failover support to ensure continuous, uninterrupted access to data. *Connection failover* provides failover protection for new connections. In more traditional scenarios, the driver fails over new connections to an alternate, or backup, database server if the primary database server is unavailable. However, Google BigQuery currently supports only the notion of a single leader node. Therefore, to ensure a continuous connection, you only need to set the ConnectionRetryCount and ConnectionRetryDelay connection properties.

See also

Connection property descriptions on page 51
ConnectionRetryCount on page 65
ConnectionRetryDelay on page 66

Timeouts

The driver supports the LoginTimeout, WSTimeout, and WSRetryCount connection properties. The LoginTimeout property specifies the amount of time, in seconds, that the driver waits for a connection to be established before timing out the connection request. In contrast, the WSTimeout property specifies the time, in seconds, that the driver waits for a response to a Web service request. The WSTimeout property can be used in conjunction with the WSRetryCount property. The WSRetryCount connection property can be used to retry select queries that have timed out.

Session Timeouts

Most remote data sources impose a limit on the duration of active sessions, meaning a session can fail with a session timeout error if the session extends past the limit. This is particularly true when connection pooling is used. The driver automatically attempts to re-establish a new session if the driver receives a session timeout error from a data source. The driver uses the initial server name, remote user ID, and remote password to re-establish the session. If the attempt fails, the driver returns an error indicating that the session timed out and the attempt to re-establish the session failed.

Web Service Request Timeouts

You can configure the driver to never time out while waiting for a response to a Web service request or to wait for a specified interval before timing out by setting the WSTimeout connection property. For fetch requests, you can configure the driver to retry the request a specified number of times by setting the WSRetryCount connection property. If subsequent attempts to retry a request fail, the driver returns an error indicating that the service request timed out and the subsequent requests failed.

See also

Connection property descriptions on page 51

Isolation levels

The driver supports the TRANSACTION_NONE isolation level because Google BigQuery does not support transactions.

Scrollable cursors

The driver supports scroll-insensitive result sets.

Note: When the driver cannot support the requested result set type or concurrency, it automatically downgrades the cursor and generates one or more SQLWarnings with detailed information.

Unicode support

Multilingual JDBC applications can be developed on any operating system using the driver to access both Unicode and non-Unicode enabled databases. Internally, Java applications use UTF-16 Unicode encoding for string data. When fetching data, the driver automatically performs the conversion from the character encoding used by the database to UTF-16. Similarly, when inserting or updating data in the database, the driver automatically converts UTF-16 encoding to the character encoding used by the database.

The JDBC API provides mechanisms for retrieving and storing character data encoded as Unicode (UTF-16) or ASCII. Additionally, the Java String object contains methods for converting UTF-16 encoding of string data to or from many popular character encodings.

Error handling

SQLExceptions

The driver reports errors to the application by throwing SQLExceptions. Each SQLException contains the following information:

- Description of the probable cause of the error, prefixed by the component that generated the error
- Native error code (if applicable)

- String containing the XOPEN SQLstate

Driver Errors

An error generated by the driver has the format shown in the following example:

```
[DataDirect][GoogleBigQuery JDBC Driver]Timeout expired.
```

You may need to check the last JDBC call your application made and refer to the JDBC specification for the recommended action.

Database Errors

An error generated by the database has the format shown in the following example:

```
[DataDirect][GoogleBigQuery JDBC Driver][GoogleBigQuery]Invalid Object Name.
```

If you need additional information, use the native error code to look up details in your database documentation.

Parameter metadata support

The driver supports returning parameter metadata as described in "Insert and Update Statements" and "Select Statements."

Insert and update statements

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

- `INSERT INTO department (col1, col2, col3) VALUES(?, ?, ?)`
- `UPDATE employee SET col1=?, col2=?, col3=? WHERE col1 operator ? [{AND | OR} col2 operator ?]`

where:

operator

is any of the following SQL operators:

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

Select statements

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

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

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

  ```
  SELECT * FROM foo WHERE bar > ?
  ```

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

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

```
SELECT * FROM foo WHERE (SELECT x FROM y    WHERE z = 1) < ?
```

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

```
SELECT col1, col2 FROM foo WHERE col1 = ? AND col2 > ?
SELECT ... WHERE colname = (SELECT col2 FROM t2 WHERE col3 = ?)
SELECT ... WHERE colname LIKE ?
SELECT ... WHERE colname BETWEEN ? and ?
SELECT ... WHERE colname IN (?, ?, ?)
SELECT ... WHERE EXISTS(SELECT ... FROM T2 WHERE col1 < ?)
```

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

```
SELECT * FROM t1 WHERE col = ? ORDER BY 1
```

Joins are supported. For example:

```
SELECT * FROM t1,t2 WHERE t1.col1 = ?
```

Fully qualified names and aliases are supported. For example:

```
SELECT a, b, c, d FROM T1 AS A, T2 AS B WHERE A.a = ? AND B.b = ?
```

ResultSet metadata support

If your application requires table name information, the Google BigQuery driver can return table name information in ResultSet metadata for Select statements. The Select statements for which ResultSet metadata is returned may contain aliases, joins, and fully qualified names. The following queries are examples of Select statements for which the ResultSetMetaData.getColumnName() method returns the correct table name for columns in the Select list:

```
SELECT id, name FROM Employee
SELECT E.id, E.name FROM Employee E
SELECT E.id, E.name AS EmployeeName FROM Employee E
SELECT E.id, E.name, I.location, I.phone FROM Employee E, EmployeeInfo I
   WHERE E.id = I.id
SELECT id, name, location, phone FROM Employee, EmployeeInfo WHERE id = empId
SELECT Employee.id, Employee.name, EmployeeInfo.location, EmployeeInfo.phone
   FROM Employee, EmployeeInfo WHERE Employee.id = EmployeeInfo.id
```

3

Connection property descriptions

You can use connection properties to customize the driver for your environment. This section lists the connection properties supported by the driver and describes each property. You can use these connection properties with either the JDBC `DriverManager` or a JDBC `DataSource`. 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, `setAuthenticationMethod("oauth2")`.
- 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: `Project={myproject }` or `Project={{myproject}}`.

The following tables provide a summary of the connection properties supported by the driver, their corresponding data source methods, and their default values.

Required properties

The following table summarizes connection properties that are required to connect to a Google BigQuery instance when the default authentication method, OAuth 2.0, is used for authentication.

Table 4: Required properties

Property	Data Source Method	Default
AccessToken on page 59	setAccessToken	None
AuthenticationMethod on page 60	setAuthenticationMethod	oauth2
ClientID on page 62	setClientID	None
ClientSecret on page 62	setClientSecret	None
Dataset on page 69	setDataset	None
Project on page 84	setProject	None
RefreshToken on page 89	setRefreshToken	None

Mapping properties

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

Table 5: Mapping properties

Property	Data Source Method	Default
ConfigOptions on page 63	setConfigOptions	VarcharLength=65535; SchemaSet=;
CreateMap on page 68	setCreateMap	notExist
SchemaMap on page 91	setSchemaMap	Default value depends on environment

OAuth 2.0 properties

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

Table 6: OAuth 2.0 properties

Property	Data Source Method	Default
AccessToken on page 59	setAccessToken	None
AuthenticationMethod on page 60	setAuthenticationMethod	oauth2
AuthURI on page 58	setAuthUri	https://accounts.google.com/o/oauth2/auth
ClientID on page 62	setClientID	None
ClientSecret on page 62	setClientSecret	None

Property	Data Source Method	Default
EnableLoginPrompt on page 70	setEnableLoginPrompt	false
RedirectURI on page 87	setRedirUri	None
RefreshToken on page 89	setRefreshToken	None
Scope on page 93	setScope	https://www.googleapis.com/auth/bigquery
TokenURI on page 102	setTokenUri	https://accounts.google.com/o/oauth2/token

Service account properties

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

Table 7: Service account properties

Property	Data Source Method	Default
AuthenticationMethod on page 60	setAuthenticationMethod	oauth2
JWTAudience on page 75	setJwtAudience	https://accounts.google.com/o/oauth2/token
ServiceAccountEmail on page 94	setServiceAccountEmail	None
ServiceAccountKeyContent on page 95	setServiceAccountKeyContent	None
ServiceAccountPrivateKey on page 96	setServiceAccountPrivateKey	None
TokenURI on page 102	setTokenUri	https://accounts.google.com/o/oauth2/token

Storage API properties

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

Property	Data Source Method	Default
StorageAPIMinPageCount on page 100	setStorageAPIMinPageCount	3
StorageAPIThreshold on page 100	setStorageAPIThreshold	10000
UseStorageAPI on page 103	setUseStorageAPI	false

Legacy SQL properties

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

Table 8: Legacy SQL properties

Property	Data Source Method	Default
AllowLargeResults on page 60	`setAllowLargeResults`	`false`
LegacyDataset on page 77	`setLegacyDataset`	`_queries_`
LegacyTable on page 78	`setLegacytable`	`_sql_*`
Syntax on page 101	`setSyntax`	`standard`

Failover properties

The following table summarizes connection properties that can be used to implement failover.

Table 9: Failover properties

Property	Data Source Method	Default
ConnectionRetryCount on page 65	`setConnectionRetryCount`	`5`
ConnectionRetryDelay on page 66	`setConnectionRetryDelay`	`1` (second)

Proxy server properties

The following table summarizes proxy server connection properties.

Table 10: Proxy Server properties

Property	Data Source Method	Default
ProxyHost on page 84	`setProxyHost`	None
ProxyPassword on page 85	`setProxyPassword`	None
ProxyPort on page 86	`setProxyPort`	`0`
ProxyUser on page 86	`setProxyUser`	None

Web Service properties

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

Table 11: Web Service properties

Property	Data Source Method	Default
WSFetchSize on page 104	`setWSFetchSize`	`10000`

Property	Data Source Method	Default
WSPoolSize on page 105	`setWSPoolSize`	`1`
WSRetryCount on page 106	`setWSRetryCount`	`5`
WSTimeout on page 107	`setWSTimeout`	`120` (seconds)

Timeout properties

The following table summarizes timeout connection properties.

Table 12: Timeout properties

Property	Data Source Method	Default
JobTimeout on page 73	`setJobTimeout`	`0` (no timeout)
LoginTimeout on page 80	`setLoginTimeout`	`0`
WSRetryCount on page 106	`setWSRetryCount`	`5`
WSTimeout on page 107	`setWSTimeout`	`120` (seconds)

Statement pooling properties

The following table summarizes statement pooling connection properties.

Table 13: Statement pooling properties

Property	Data Source Method	Default
ImportStatementPool on page 72	`setImportStatementPool`	None
MaxPooledStatements on page 82	`setMaxPooledStatements`	`0`
RegisterStatementPoolMonitorMBean on page 90	`setRegisterStatementPoolMonitorMBean`	`false`

Additional properties

The following table summarizes additional connection properties.

Table 14: Additional properties

Property	Data Source Method	Default
BinaryLength on page 61	`setBinaryLength`	`65535`
ConvertNull on page 67	`setConvertNull`	`1`
EnableCatalogSupport on page 69	`setEnableCatalogSupport`	`false`

Property	Data Source Method	Default
FetchSize on page 71	setFetchSize	100 (rows)
JavaDoubleToString on page 73	setJavaDoubleToString	false
JsonFormat on page 74	setJsonFormat	raw
Location on page 78	setLocation	None
LogConfigFile on page 79	setLogConfigFile	ddlogging.properties
MaximumBytesBilled on page 81	setMaximumBytesBilled	0 (no limit)
MaximumBillingTier on page 80	setMaximumBillingTier	0
PrimaryKeyPattern on page 83	setPrimaryKeyPattern	*
RefreshSchema on page 87	setRefreshSchema	true
RefreshSchemaForDDL on page 88	setRefreshSchemaForDDL	true
RetryExceptions on page 90	setRetryExceptions	false
ServerName on page 94	setServerName	www.googleapis.com
SpyAttributes on page 97	setSpyAttributes	None
UseStreamingInsert on page 104	useStreamingInsert	true
UseQueryCache on page 102	setUseQueryCache	1

For details, see the following topics:

- AuthURI
- AccessToken
- AllowLargeResults
- AuthenticationMethod
- BinaryLength
- ClientID
- ClientSecret
- ConfigOptions
- ConnectionRetryCount
- ConnectionRetryDelay
- ConvertNull

- CreateMap
- Dataset
- EnableCatalogSupport
- EnableLoginPrompt
- FetchSize
- ImportStatementPool
- JavaDoubleToString
- JobTimeout
- JsonFormat
- JWTAudience
- KMSKeyName
- LegacyDataset
- LegacyTable
- Location
- LogConfigFile
- LoginTimeout
- MaximumBillingTier
- MaximumBytesBilled
- MaxPooledStatements
- PrimaryKeyPattern
- Project
- ProxyHost
- ProxyPassword
- ProxyPort
- ProxyUser
- RedirectURI
- RefreshSchema
- RefreshSchemaForDDL
- RefreshToken
- RegisterStatementPoolMonitorMBean
- RetryExceptions
- SchemaMap
- Scope

AuthURI

Purpose

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

Valid Values

String

where:

String

is the endpoint for retrieving the OAuth 2.0 authorization code from a private endpoint.

Notes

- When this endpoint is queried, the authorization service presents an interface prompting the user to approve or deny access to backend data.
- See "Configuring OAuth 2.0 authentication" for examples and more information.

Data Source Method

`setAuthUri`

Default

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

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

AccessToken

Purpose

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

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

Valid Values

`string`

where:

`string`

is the access token you have obtained from Google BigQuery.

Notes

- If a value for the AccessToken property is not specified, the driver uses the value of the RefreshToken property to generate an access token to make a connection.
- If both AccessToken and RefreshToken values are not specified, the driver cannot make a successful connection.
- If both AccessToken and RefreshToken values are specified, the driver uses the AccessToken value. However, if the AccessToken value expires, it uses the RefreshToken value to generate a new AccessToken value.

Data Source Method

`setAccessToken`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Generating access tokens and refresh tokens on page 42

RefreshToken on page 89

Required properties

AllowLargeResults

Purpose

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

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

Valid Values

`false` | `true`

Behavior

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

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

Data Source Method

`setAllowLargeResults`

Default

`false`

Data Type

Boolean

See also

LegacyDataset on page 77

LegacyTable on page 78

Legacy SQL properties

AuthenticationMethod

Purpose

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

Valid values

`oauth2` | `serviceaccount`

Behavior

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

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

Data Source Method

`setAuthenticationMethod`

Default

`oauth2`

Data Type

String

See Also

Configuring OAuth 2.0 authentication on page 39

Configuring service account authentication on page 41

BinaryLength

Purpose

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

Valid Values

x

where:

x

is a positive integer between `1` and `2147483647`.

Behavior

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

Data Source Method

`setBinaryLength`

Default

`65535`

Data Type

Int

See also

Additional properties

ClientID

Purpose

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

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

Valid Values

`string`

where:

`string`

is the consumer key for your application.

Data Source Method

`setClientID`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Generating access tokens and refresh tokens on page 42

Required properties

ClientSecret

Purpose

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

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

Valid Values

string

where:

string

is the consumer secret for your aplication.

Data Source Method

`setClientSecret`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Generating access tokens and refresh tokens on page 42

Required properties

ConfigOptions

Purpose

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

Notes

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

Valid Values

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

where:

key

is one of the following configuration options:

- `VarcharLength`

- `SchemaSet`

`value`

specifies a setting for the configuration option.

When specifying configuration options in a connection string, key value pairs must be enclosed in parentheses and separated by a semicolon. For example:

```
ConfigOptions=(VarcharLength=65535;SchemaSet=;)
```

Data Source Method

`setConfigOptions`

Default

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

Data Type

String

See also

Mapping properties

SchemaSet (Configuration Option) on page 64

VarcharLength (Configuration Option) on page 65

SchemaSet (Configuration Option)

Purpose

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

Valid Values

`String` | `*`

where:

`String`

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

Behavior

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

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

Notes

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

Default

The project and dataset specified at connection.

See also

VarcharLength (Configuration Option) on page 65

VarcharLength (Configuration Option)

Purpose

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

Valid Values

x

where:

x

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

Default

65535

See also

SchemaSet (Configuration Option) on page 64

ConnectionRetryCount

Purpose

The number of times the driver retries connection attempts to a Google BigQuery instance until a successful connection is established.

Valid Values

`0` | *x*

where *x* is a positive integer that represents the number of retries.

Behavior

If set to `0`, the driver does not try to reconnect after the initial unsuccessful attempt.

If set to *x*, the driver retries connection attempts the specified number of times. If a connection is not established during the retry attempts, the driver returns an exception that is generated by the last database server to which it tried to connect.

Example

If this property is set to `2`, the driver retries the server twice after the initial retry attempt.

Notes

- If an application sets a login timeout value (for example, using DataSource.loginTimeout or DriverManager.loginTimeout), and the login timeout expires, the driver ceases connection attempts.
- The ConnectionRetryDelay property specifies the wait interval, in seconds, to occur between retry attempts.

Data Source Method

`setConnectionRetryCount`

Default

`5`

Data Type

int

See also

Failover properties

ConnectionRetryDelay

Purpose

The number of seconds the driver waits between connection retry attempts when ConnectionRetryCount is set to a positive integer.

Valid Values

`0` | *x*

where:

x

is a number of seconds.

Behavior

If set to `0`, the driver does not delay between retries.

If set to *x*, the driver waits between connection retry attempts the specified number of seconds.

Example

If ConnectionRetryCount is set to 2 and this property is set to 3, the driver retries the server twice after the initial retry attempt. The driver waits 3 seconds between retry attempts.

Data Source Method

`setConnectionRetryDelay`

Default

1 (second)

Data Type

int

See also

Failover properties

ConvertNull

Purpose

Controls how data conversions are handled for null values.

Valid Values

`0 | 1`

Behavior

If set to `0`, the driver does not perform the data type check if the value of the column is null. This allows null values to be returned even though a conversion between the requested type and the column type is undefined.

If set to `1`, the driver checks the data type being requested against the data type of the table column that stores the data. If a conversion between the requested type and column type is not defined, the driver generates an "unsupported data conversion" exception regardless of whether the column value is NULL.

Data Source Method

`setConvertNull`

Default

`1`

Data Type

int

See also

Additional properties

CreateMap

Purpose

Specifies whether the driver creates a new map of the Google BigQuery data model when establishing the connection.

Valid Values

`forceNew` | `notExist` | `no` | `session`

Behavior

If set to `forceNew`, the driver deletes the current schema map specified by the SchemaMap property and creates a new one at the same location.

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

If set to `notExist`, the driver uses the current schema map specified by the SchemaMap property. If one does not exist, the driver creates one.

If set to `no`, the driver uses the current schema map specified by the SchemaMap property. If one does not exist, the connection fails.

If set to `session`, the driver does not create or use schema map. It stores metadata in the memory.

Data Source Method

`setCreateMap`

Default

`notExist`

Data Type

String

See Also

- SchemaMap on page 91
- Mapping properties

Dataset

Purpose

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

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

```
project.dataset.table
```

Valid Values

string

where:

string

is the name of your Google BigQuery dataset.

Data Source Method

`setDataset`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Project on page 84

Required properties

EnableCatalogSupport

Purpose

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

Valid values

`true` | `false`

Behavior

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

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

Notes

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

Data Source Method

`setEnableCatalogSupport`

Default

`false`

Data Type

String

See Also

SchemaSet (Configuration Option) on page 64

EnableLoginPrompt

Purpose

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

Valid values

`true` | `false`

Behavior

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

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

Notes

- This property is used only for the dynamic authorization code grant flow. See "Dynamic authorization code grant" for a full list of requirements.

Data Source Method

`setEnableLoginPrompt`

Default

`false`

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 and WSFetchSize 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 Method

`setFetchSize`

Default

`100` (rows)

Data Type

Int

See also

Additional properties

ImportStatementPool

Purpose

Specifies the path and file name of the file to be used to load the contents of the statement pool. When this property is specified, statements are imported into the statement pool from the specified file.

Valid Values

`string`

where:

`string`

is the path and file name of the file to be used to load the contents of the statement pool.

Notes

- If the driver cannot locate the specified file when establishing the connection, the connection fails and the driver throws an exception.
- For more information, refer to "Statement Pool Monitor" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Method

`setImportStatementPool`

Default

empty string

Data Type

String

JavaDoubleToString

Purpose

Determines which algorithm the driver uses when converting a double or float value to a string value. By default, the driver uses its own internal conversion algorithm, which improves performance.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver uses the JVM algorithm when converting a double or float value to a string value. If your application cannot accept rounding differences and you are willing to sacrifice performance, set this value to `true` to use the JVM conversion algorithm.

If set to `false`, the driver uses its own internal algorithm when converting a double or float value to a string value. This value improves performance, but slight rounding differences within the allowable error of the double and float data types can occur when compared to the same conversion using the JVM algorithm.

Data Source Method

`setJavaDoubleToString`

Default

`false`

Data Type

boolean

See also

Additional properties

JobTimeout

Purpose

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

Valid Values

`0` | *x*

where:

x

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

Behavior

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

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

Data Source Method

`setJobTimeout`

Default

`0` (no timeout)

Data Type

Int

See also

Timeout properties

JsonFormat

Purpose

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

Valid Values

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

Behavior

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

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

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

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

Examples

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

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

Data Source Method

`setJsonFormat`

Default

`raw`

Data Type

String

See also

Additional properties

JWTAudience

Purpose

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

Valid Values

`String`

where:

`String`

is the JWT audience claim associated with your service account.

Data Source Methods

`setJwtAudience`

Default Value

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

Data Type

String

See also

Configuring service account authentication on page 41

KMSKeyName

Purpose

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

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

Valid Values

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

where:

`project`

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

`location`

specifies the geographical location where your dataset is stored.

`keyring`

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

`key`

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

Notes

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

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

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

Data Source Method

`setKMSKeyName`

Default

None

Data Type

String

LegacyDataset

Purpose

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

Valid Values

`string`

where:

`string`

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

Data Source Method

`setLegacyDataset`

Default

`_queries_`

Data Type

String

See also

AllowLargeResults on page 60

LegacyTable on page 78

Legacy SQL properties

LegacyTable

Purpose

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

Valid Values

`string`

where:

`string`

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

Data Source Method

`setLegacytable`

Default

`_sql_*`

Data Type

String

See also

AllowLargeResults on page 60

LegacyDataset on page 77

Legacy SQL properties

Location

Purpose

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

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

Valid Values

`string`

where:

string

is the geographical location where your dataset is stored.

Data Source Method

`setLocation`

Default

None

Data Type

String

See also

Dataset on page 69

Additional properties

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 Method

`setLogConfigFile`

Default

`ddlogging.properties`

Data Type

String

LoginTimeout

Purpose

The amount of time, in seconds, that the driver waits for a connection to be established before timing out the connection request.

Valid Values

`0` | *x*

where:

x

is a positive integer that represents a number of seconds.

Behavior

If set to `0`, the driver does not time out a connection request.

If set to *x*, the driver waits for the specified number of seconds before returning control to the application and throwing a timeout exception.

Data Source Method

`setLoginTimeout`

Default

`0`

Data Type

int

See also

Timeout properties

MaximumBillingTier

Purpose

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

Valid Values

x

where:

x

is a positive integer that identifies the tier.

Data Source Method

`setMaximumBillingTier`

Default

`0`

Data Type

Int

See also

Additional properties

MaximumBytesBilled

Purpose

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

Valid Values

`0` | *x*

where:

x

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

Behavior

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

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

Data Source Method

`setMaximumBytesBilled`

Default

`0` (no limit)

Data Type

Int

See also

Additional properties

MaxPooledStatements

Purpose

The maximum number of pooled prepared statements for this connection. Setting MaxStatements to an integer greater than zero (0) enables the driver's internal prepared statement pooling, which is useful when the driver is not running from within an application server or another application that provides its own prepared statement pooling.

Valid Values

`0` | *x*

where

x

is a positive integer that represents a number of pooled prepared statements.

Behavior

If set to `0`, the driver's internal prepared statement pooling is not enabled.

If set to *x*, the driver enables the DataDirect Statement Pool and uses the specified value to cache a certain number of prepared statements created by an application. If the value set for this property is greater than the number of prepared statements that are used by the application, all prepared statements that are created by the application are cached. Because CallableStatement is a sub-class of PreparedStatement, CallableStatements also are cached.

Example

If the value of this property is set to `20`, the driver caches the last 20 prepared statements that are created by the application.

Data Source Method

`setMaxPooledStatements`

Default

`0`

Data Type

int

See also

Statement pooling properties

PrimaryKeyPattern

Purpose

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

Valid Values

`*` | *`column_name`*

where

`column_name`

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

Behavior

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

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

Data Source Method

`setPrimaryKeyPattern`

Default

`*`

Data Type

String

See also

Mapping properties

Project

Purpose

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

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

```
project.dataset.table
```

Valid Values

string

where:

string

is the name of your Google BigQuery project.

Data Source Method

`setProject`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Dataset on page 69

Required properties

ProxyHost

Description

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 Method

`setProxyHost`

Default

empty string

See also

- Connecting through a proxy server on page 38
- Proxy server properties

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 Method

`setProxyPassword`

Default

empty string

See also

Connecting through a proxy server on page 38

Proxy server properties

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 Method

`setProxyPort`

Default

`0`

See also

Connecting through a proxy server on page 38

Proxy server properties

ProxyUser

Purpose

Specifies the 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 Method

`setProxyUser`

Default

empty string

See also

Connecting through a proxy server on page 38

Proxy server properties

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

None

Data Type

String

RefreshSchema

Purpose

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

Valid Values

`true` | `false`

Behavior

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

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

Notes

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

Data Source Method

`setRefreshSchema`

Default

`true`

See also

Additional properties

RefreshSchemaForDDL

Purpose

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

Valid Values

`true` | `false`

Behavior

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

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

Notes

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

Data Source Method

`setRefreshSchemaForDDL`

Default

`true`

See also

Additional properties

RefreshToken

Purpose

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

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

Valid Values

`string`

where:

`string`

is the refresh token you have obtained from Google BigQuery.

Notes

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

Data Source Method

`setRefreshToken`

Default

None

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Generating access tokens and refresh tokens on page 42

AccessToken on page 59

Required properties

RegisterStatementPoolMonitorMBean

Purpose

Registers the Statement Pool Monitor as a JMX MBean when statement pooling has been enabled with MaxPooledStatements. This allows you to manage statement pooling with standard JMX API calls and to use JMX-compliant tools, such as JConsole.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver registers an MBean for the statement pool monitor for each statement pool. This gives applications access to the Statement Pool Monitor through JMX when statement pooling is enabled.

If set to `false`, the driver does not register an MBean for the Statement Pool Monitor for any statement pool.

Notes

Registering the MBean exports a reference to the Statement Pool Monitor. The exported reference can prevent garbage collection on connections if the connections are not properly closed. When garbage collection does not take place on these connections, out of memory errors can occur.

Data Source Method

setRegisterStatementPoolMonitorMBean

Default

`false`

Data Type

Boolean

See also

- Refer to "Statement Pool Monitor" in the *Progress DataDirect for JDBC Drivers Reference* for further details.
- MaxPooledStatements on page 82
- Statement pooling properties

RetryExceptions

Purpose

Determines whether the driver retries an API call execution when an HTTP failure or driver exception occurs. The number of retry attempts is specified by the WSRetryCount property. By default, exceptions caused during a call execution cannot be retried.

Valid Values

`true` | `false`

Behavior

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

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

Data Source Method

`setRetryExceptions`

Default

`false`

See also

Additional properties

SchemaMap

Purpose

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

Valid Values

`string`

where:

`string`

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

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

Notes

- When connecting to a Google BigQuery instance, the driver looks for the schema map configuration file. If the configuration file does not exist, the driver creates the schema map configuration file using the name and location you have provided. If you do not provide a name and location for the configuration file, the driver creates it using default values.
- The driver uses the path specified in this connection property to store additional internal files.

Example

As the following examples show, escapes are needed when specifying SchemaMap for a data source but are not used when specifying SchemaMap in a `DriverManager` connection URL.

Driver Manager Example

```
jdbc:datadirect:googlebigquery:Project=myproject;Dataset=mydataset;
AccessToken=abcdefghi12345678;RefreshToken=wxyz123456789;
ClientID=123abc.apps.googleusercontent.com;ClientSecret=ab123xy;
SchemaMap=C:\Users\Default\AppData\Local\Progress\DataDirect\
GoogleBigQuery_Schema\abc@defcorp.com.config
```

Data Source Example

```
GoogleBigQueryDataSource ds = new GoogleBigQueryDataSource();
ds.setDescription("My Google BigQuery Datasource");
ds.setProject("myproject");
ds.setDataset("mydataset");
ds.setAccessToken("abcdefghi12345678");
ds.setRefreshToken("wxyz123456789");
ds.setClientID("123abc.apps.googleusercontent.com");
ds.setClientSecret("ab123xy");
ds.setSchemaMap("C:\\Users\\Default\\AppData\\Local\\Progress
\\DataDirect\\GoogleBigQuery_Schema\\abc@defcorp.com.config")
```

Data Source Method

`setSchemaMap`

Default

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

- Windows
  - DD_HOME environment variable
  - dd.home system property
  - LOCALAPPDATA environment variable
  - APPDATA environment variable
  - user.home system property

  For Windows, the file path takes the following format:

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

- UNIX/Linux
  - DD_HOME environment variable
  - dd.home system property

- user.home system property

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

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

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Mapping properties

Scope

Purpose

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

Valid Values

string

where:

string

is a space-separated list of security scopes.

Examples

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

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

Data Source Method

setScope

Default

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

Data Type

String

See Also

- AccessToken on page 59

- Configuring OAuth 2.0 authentication on page 39
- OAuth 2.0 properties

ServerName

Purpose

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

Valid Values

url

where:

url

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

Notes

- The HostName property is an alias for the ServerName property.

Data Source Method

`setServerName`

Default

`www.googleapis.com`

Data Type

String

ServiceAccountEmail

Purpose

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

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

Valid Values

string

where:

string

is your service account's email address.

Data Source Method

`setServiceAccountEmail`

Default

None

Data Type

String

See also

Configuring service account authentication on page 41

ServiceAccountPrivateKey on page 96

Service account properties

ServiceAccountKeyContent

Purpose

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

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

Valid Values

string

where:

string

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

Notes

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

Example

As shown in this example, surrounding quotation marks should be omitted when specifying the value.

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

Data Source Method

`setServiceAccountKeyContent`

Default

None

Data Type

String

See also

Configuring service account authentication on page 41

ServiceAccountEmail on page 94

ServiceAccountPrivateKey on page 96

Service account properties

ServiceAccountPrivateKey

Purpose

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

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

Valid Values

`string`

where:

`string`

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

Notes

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

Data Source Method

`setServiceAccountPrivateKey`

Default

None

Data Type

String

See also

Configuring service account authentication on page 41

ServiceAccountEmail on page 94

ServiceAccountKeyContent on page 95

Service account properties

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`

`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`.

Attribute	Description
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.

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*.

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)
```

Data Source Method

setSpyAttributes

Default

Empty string

Data Type

String

StorageAPIMinPageCount

Purpose

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

Valid Values

x

where:

x

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

Data Source Method

`setStorageAPIMinPageCount`

Default

3 (pages)

Data Type

Int

See also

- Storage API properties
- UseStorageAPI on page 103
- StorageAPIThreshold on page 100

StorageAPIThreshold

Purpose

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

Valid Values

x

where:

x

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

Data Source Method

`setStorageAPIThreshold`

Default

`10000` (rows)

Data Type

Int

See also

- Storage API properties
- UseStorageAPI on page 103
- StorageAPIMinPageCount on page 100

Syntax

Purpose

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

Valid Values

`standard` | `legacy`

Behavior

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

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

Data Source Method

`setSyntax`

Default

`standard`

Data Type

String

See also

Standard and legacy SQL support on page 15

Legacy SQL properties

TokenURI

Purpose

Specifies the endpoint for retrieving access tokens when either OAuth 2.0 or service account authentication is enabled.

Valid Values

String

where:

String

is the endpoint used to retrieve access tokens.

Notes

See "Configuring OAuth 2.0 authentication" and "Configuring service account authentication" for more information.

Data Source Method

`setTokenUri`

Default Value

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

Data Type

String

See also

Configuring OAuth 2.0 authentication on page 39

Configuring service account authentication on page 41

UseQueryCache

Purpose

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

Valid Values

`true` | `false`

Behavior

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

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

Data Source Method

`setUseQueryCache`

Default

`true`

Data Type

Boolean

See also

Additional properties

UseStorageAPI

Purpose

Specifies whether the driver uses the Google BigQuery Storage API when fetching large result sets based on the values of the StorageAPIThreshold and StorageAPIMinPageCount connection properties.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver uses the Storage API for selects when the number of rows in the result set exceeds the value of the StorageAPIThreshold property, and the number of pages in the result set exceeds the value of the StorageAPIMinPageCount property.

If set to `false`, the driver does not use the Storage API, and the StorageAPIThreshold and StorageAPIMinPageCount properties are ignored.

Data Source Method

`setUseStorageAPI`

Default

`false`

Data Type

Boolean

See also

- Storage API properties
- StorageAPIMinPageCount on page 100
- StorageAPIThreshold on page 100

UseStreamingInsert

Purpose

Determines whether the driver uses Google BigQuery's streaming API (InsertAll) to execute batch inserts. This improves performance for INSERT operations.

Valid Values

`true` | `false`

Behavior

If set to `true`, the driver uses the streaming API for batch inserts.

If set to `false`, the driver executes row by row inserts.

Notes

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

Data Source Method

`setUseStreamingInsert`

Default

`true`

Data Type

Boolean

See also

Google BigQuery Streaming API on page 46

Performance considerations on page 38

WSFetchSize

Purpose

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

Valid Values

`0` | *x*

where:

x

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

Behavior

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

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

Notes

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

Data Source Method

`setWSFetchSize`

Default

`1000000`

Data Type

Int

See also

Web Service properties

WSPoolSize

Purpose

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

Valid Values

x

where:

x

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

Notes

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

Data Source Method

`setWSPoolSize`

Default

`1`

Data Type

Int

See also

Web Service properties

WSRetryCount

Description

The number of times the driver retries a timed-out Select, Insert, Update, or Delete 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 Method

`setWSRetryCount`

Default

`5`

Data Type

Int

See also

Web Service properties

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 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 Method

`setWSTimeout`

Default

`120` (seconds)

Data Type

Int

See also

Web Service properties