Progress DataDirect for ODBC for Snowflake

User's Guide

Release 8.0.1

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Snowflake Driver

The Progress® DataDirect® for ODBC™ for Snowflake™ driver leverages the SQL engine functionality of Snowflake to execute SQL queries. It supports seamless integration with third-party software. It uses data encryption for security.

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

For details, see the following topics:

- Troubleshooting
- Contacting Technical Support

What's new in this release?

Support and Certifications

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

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

Changes for 8.0.1 GA

- **Enhancements**:
  - The driver has been enhanced to support key-pair authentication. When key-pair authentication is enabled (`AuthenticationMethod=21`), you can authenticate to Snowflake using a pair of private and public keys. To configure the driver to use key-pair authentication, you can use the new Private Key Content (PrivateKeyContent), Private Key File (PrivateKeyFile), and Private Key Passphrase (PrivateKeyPassphrase) connection options. For details, see Key-pair authentication on page 46.
  - The ODBC Setup Dialog on Windows has been replaced with the Snowflake Configuration Manager. As with the ODBC Setup Dialog, the Configuration Manager is available through the ODBC Administrator. The Configuration Manager enables you to configure data sources, generate connection strings, test connections, and retrieve OAuth tokens. See the following topics for details:
    - Configuring data sources with the Configuration Manager on page 33
    - Generating connection strings with the Configuration Manager on page 34
    - Testing connections and queries with the Configuration Manager on page 41
    - Obtaining access and refresh tokens using the Configuration Manager on page 53
  - The driver has been enhanced to support the authorization code grant, the client credentials grant, and the refresh token grant, in addition to the access token flow. Also, on Windows, the Snowflake Configuration Manager may now be used to obtain access and refresh tokens. See OAuth 2.0 authentication on page 48 for details.
  - The driver has been modified to map Snowflake fixed-point number types where precision and scale cannot be modified to the Numeric data type by default. In addition, the Integer Field Mapping (IntegerFieldMapping) connection option has been added to the driver. This connection option allows you to map these fixed-point number types to BigInt. For details, see Integer Field Mapping on page 88 and Data types on page 21.
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The driver now allows you to specify whether the driver fetches metadata for only tables in the database to which you are connected when a database name is not specified in metadata calls. When enabled, this behavior can provide better performance for metadata calls by reducing the number of tables queried. You can use the new Use Session Database For Metadata (UseSessionDatabaseForMetadata) connection option to configure the driver's behavior. See Use Session Database for Metadata on page 104 for details.

- The driver has been enhanced to support connecting to Snowflake partner applications. You can use the new Partner Application Name (PartnerApplicationName) connection option to specify the name of the partner application to which you want to connect. See Partner Application Name on page 92 for details.
- The driver has been enhanced to fall back to JSON query format when the arrow format is not properly initialized. The driver uses a high-speed arrow transfer that requires the restricted APIs from java.nio package. When the JVM is not correctly configured for Java SE 16 and higher, an exception is returned when the driver executes a query. To allow you to continue executing queries in this scenario, you can configure the new Arrow Fallback To JSON (`ArrowFallbackToJson`) option to switch to the JSON query format. See Arrow Fallback To JSON on page 80 for details.
- A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and `odbc.ini` files. At connection, the driver decrypts these passwords and passes them to the data source as required. See Password Encryption Tool (UNIX/Linux only) on page 42 for details.
- The driver supports using the COPY command for loading and unloading data from local file systems and cloud platforms, such as Amazon S3, Google Cloud, and Microsoft Azure. See Using COPY command for details.

- **Changed Behavior**:
  - The Authentication Method (Authenticator) option has been updated:
    - The attribute, Authenticator, has changed to AuthenticationMethod. To support existing configurations, the original attribute name will continue to be supported for this release.
    - The valid values have changed from enum values to numeric values.

    See Authentication Method on page 82 for details.
  - The names and attributes for the following options have been changed:
    - The Role (Role) option has been renamed Role Name (RoleName)
    - The Token (Token) option has been renamed Access Token (AccessToken).
    - The User (LogonID) option has been renamed User (User).

    To support existing configurations, the original attribute names for these options will continue to be supported for this release.
  - The following options are no longer exposed because they are not relevant for Snowflake connections:
    - Application Using Threads (ApplicationUsingThreads)
    - Fetch Size (FetchSize)
    - IANAAppCodePage
  - The valid values for Client Session Keep Alive (ClientSessionKeepAlive) have changed from boolean to numeric. See Client Session Keep Alive on page 84 for details.
  - The default value for the SQL Engine Mode (SQLEngineMode) has changed from `0` (Auto) to `1` (Server) on Windows platforms. See SQL Engine Mode on page 102 for details.
  - The installer program now requires you to install a JRE that is Java SE 11 or higher before running the installer. In earlier versions, the JRE used by the installer program was included in the product. However, to avoid potential security vulnerabilities, the installer program no longer includes a JRE. Instead, the installer program uses the JRE in your environment to allow for the most secure version of a JRE to be used.

Note:

- This change does not affect the JVM requirements for the driver. For the latest driver requirements, refer to the Product Compatibility Guide:

  https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf

- The installer program cannot remove the already installed JRE files from the install directory automatically. Remove them manually.
- To install the 32-bit drivers on 64-bit Windows platforms and start the SQL engine service, the installer requires the 32-bit version of Java SE 11 or higher installed on your machine and defined on your path.

- The installer program has been updated to no longer install common components for drivers that do not use them. As a result, the installer program no longer installs the following common components for the Snowflake driver:
  - OpenSSL library files, such as ivopenssl30.so, ddopenssl30.so, fips.so, and openssl.cnf
  - Curl library files, such as libivcurl28.so and libddcurl28.so

  If the files are not used by any other DataDirect driver on your machine, you can safely remove them from existing product directories.
- For Linux platforms, the product package no longer includes the ODBC Cursor library file (odbccurs.so) because it has some known security vulnerabilities that could potentially expose you to security risks.

Note: The installer program cannot remove the ODBC Cursor library file automatically while installing a new version of the driver. Remove it manually.

Highlights of 8.0.0 Release

- The driver serves as a complete pass-through driver. It leverages the Snowflake SQL engine to execute queries.
- The driver provides enhanced performance over the native ODBC Snowflake driver.
- The driver supports create, read, update, and delete operations.
- The driver provides proxy support. See Proxy server support on page 60 and Connection option descriptions on page 73 for more information.

Driver requirements

Data source and platform requirements

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

Java requirements

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

Installing and setting up the driver (Windows)

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

To begin accessing data with the driver:

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

`PROGRESS_DATADIRECT_ODBC_nn_WIN_xx_INSTALL.exe`

c) Follow the prompts to complete installation.

Note:

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

---

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

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

---

4. Open the Snowflake Configuration Manager through the **User DSN** or **System DSN** tab.
   - **User DSN**: If you installed a default DataDirect ODBC user data source as part of the installation, select the appropriate data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.
   - **System DSN**: To configure a new system data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager in your browser.

Note: Configuring data sources using the **File DSN** tab in the ODBC Administrator is not currently supported.

---

5. In the Configuration Manager **Connection** tab, provide values for the following connection options; then, click **Apply**:

   For user ID and password authentication:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Account Name**: Type the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
   - **Database Name**: Type the name of database to which you want to connect.
   - **Password**: Type the password that is used to connect to the Snowflake instance.
   - **Schema**: Type the schema for the specified database.
   - **User**: Type the user name that is used to connect to the Snowflake instance.
   - **Warehouse**: Type the name of the virtual warehouse.
   - **Role Name**: Type the role for access control in the Snowflake session initiated by the driver.

Note: User and Password connection options can also be specified in the logon dialog box or passed by your application.

Note: The driver supports a number of authentication methods. See Authentication on page 43 for more information.

6. Set the values for any additional connection options that you want to configure. To view more options, click on the tabs on the dialog. See the following resources for additional information on optional features and functionality:
   - Connection string examples on page 18 provides connection string examples that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
   - Connection option descriptions on page 73 provides a complete list of supported options by functionality.
   - Configuring data sources with the Configuration Manager on page 33 guides you through using the GUI to configure the driver.
   - Performance considerations on page 62 describes connection options that affect performance, along with recommended settings.

   Note: For most connections, specifying the minimum required connection options is sufficient to begin accessing data; however, you can provide values for optional connection options to use additional supported features and improve performance.
7. Click **Test Connect** to attempt to connect to the data source using the connection options.
8. The logon dialog appears. If not already specified, update the following fields; then, click **OK**.
   - **User**: Type the user name that is used to connect to a Snowflake database. For example, `jsmith`.
   - **Password**: Type the password that is used to connect to the instance.

   Note: The information you enter in the logon dialog box during a test connect is not saved.
9. If the test was successful, the window displays a confirmation message.
10. Click **Save** to save the data source definition. The values you have specified are the defaults used when you connect to the data source. You can change these defaults by reopening the Configuration Manager to modify your data source, or you can override these defaults by connecting to the data source using a connection string with alternate values.
11. Connect to your instance and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:
    - Example Application: The example application allows you to test connect, execute SQL statements, and practice using the ODBC API right out of the box.
    - Microsoft Excel: Excel is a spreadsheet tool that allows you to connect, view tables, and execute SQL statements against your data.
    - Connection option descriptions on page 73: This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

See also

Using a connection string on page 35

Installing and setting up the driver (Linux)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_nn_LINUX_xx_INSTALL.bin`

   c) Follow the prompts to complete installation.

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

2. Configure the environment variables:
   a) Check your permissions. You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.
   b) Run one of the following product setup scripts from the installation directory to set variables: `odbc.sh` or `odbc.csh`. For Korn, Bourne, and equivalent shells, execute `odbc.sh`. For a C shell, execute `odbc.csh`. Executing the setup script:

      - Sets the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For details, see "ODBCINI."
      - Sets the library path environment variable for your Linux operating system, `LD_LIBRARY_PATH`, to include the directory containing your JVM's `libjvm.so` file. For details, see "Library search path."

3. Configure the driver using one of the following methods:

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

User ID and Password Authentication

```
[ODBC Data Sources]
Snowflake=DataDirect 8.0 Snowflake


[Snowflake]
Driver=ODBCHOME/lib/xxsnowflake.yy
...
AccountName=MyAccountName.us-east-1
...
DatabaseName=MyDB
...
DataSourceName=MyDSN
...
RoleName=MyRole
...
Schema=MySchema
...
Warehouse=MyWH
...
```

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

See Configuration through the system information (odbc.ini) file on page 36 for more information.

- **Connection string**: The driver also supports using connection strings for DSN (data source name), File DSN, or DSN-less connections. See Using a connection string on page 35, DSN-less connections, for more information. For examples, see Connection string examples on page 18.

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

---

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

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

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

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

5. Connect to your instance and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:

   - Example Application: The example application is a command-line tool that allows you to test connect, execute SQL statements, and practice using the ODBC API in environments that do not support GUIs.

This completes the deployment of the driver.

See also

ODBCINI on page 31
Library search path on page 30
Connection string examples on page 18

Connection string examples

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

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

See the following sections for connection string examples.

- User ID and password authentication
- Browser-based authentication
- Key-pair authentication
- OAuth 2.0 authentication
- Proxy server

User ID and password authentication

This string includes the options used to connect using basic user name and password authentication.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;User=JohnQPublic;
   Password=SECRET;
```

For more information on these options and values, see User ID and password authentication on page 44.

Browser-based SSO authentication

This string includes the options used to connect using browser-based SSL authentication.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   AuthenticationMethod=45;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;
   User=JohnQPublic;
```

For more information on these options and values, see Browser-based SSO authentication on page 45.

Key-pair authentication

This string includes the options used to connect using key-pair authentication.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   AuthenticationMethod=21;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;
   User=JohnQPublic;PrivateKeyFile=C:\Program Files\privatekey.p8;
   PrivateKeyPassphrase=abc123;
```

For more information on these options and values, see Key-pair authentication on page 46.

OAuth 2.0 authentication

This string includes the options required for the OAuth 2.0 refresh token grant.

```
DRIVER=DataDirect 8.0 Snowflake;AuthenticationMethod=24;
   AccountName=account_name.us-east-1;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;

   ClientId=cd34efg5678h9ij87klm6543no32pqr10st987;
   ClientSecret=098zyx765wvu432tsr123qpo456;
   TokenURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/token-request;
   RefreshToken=abc12cd34efg5678h9ij87klm6543no32pqr10;
```

For more information on these options and values, see OAuth 2.0 authentication on page 48.

Proxy server

This string includes the options used to connect through a proxy server using user ID and password authentication.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;User=JohnQPublic;
   Password=SECRET;ProxyHost=pserver;ProxyPassword=proxys3cr3t;
   ProxyPort=1234;ProxyUser=jsmith;
```

For more information on these options and values, see Proxy server support on page 60.

See also

Using a connection string on page 35

Version string information

The driver for Snowflake has a version string of the format:

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

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

XX.YY.ZZZZ(U*BBBB*)

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

XX.YY.ZZZZ

where:

XX is the major version of the product.

YY is the minor version of the product.

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

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

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

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

For example:

```
08.00.0024 (B0547, U0390, J000001)
|________|  |___|  |___|  |_____|
  Driver     Bas    Utl    Java
```

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

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

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

The syntax for the tool is:

```
ivtestlib shared_object
```

or

```
ddtestlib shared_object
```

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

```
ivtestlib ivsnowflake28.so
```

returns:

```
08.00.0024 (B0002, U0001, J000003)
```

For example, for the Driver Manager on Linux:

```
ivtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

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

```
ddtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

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

```
ivtestlib libivicu28.so
08.00.0001
```

Note: On Linux, the full path to the driver does not have to be specified for the test loading tool.

getFileVersionString function

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

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

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

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

Data types

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

Table 1: Snowflake Data Types

Snowflake Data Type	ODBC Data Type
ARRAY	SQL_VARCHAR
BIGINT[1]	SQL_NUMERIC
BINARY	SQL_BINARY
BOOLEAN	SQL_BIT
CHAR	SQL_VARCHAR
DATE	SQL_TYPE_DATE
DECIMAL	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
FLOAT	SQL_DOUBLE
INTEGER[1]	SQL_NUMERIC
NUMBER	SQL_DECIMAL
OBJECT	SQL_VARCHAR
REAL	SQL_DOUBLE
TIME	SQL_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP
TIMESTAMP_LTZ	SQL_TYPE_TIMESTAMP
TIMESTAMP_NTZ	SQL_TYPE_TIMESTAMP
TIMESTAMP_TZ	SQL_TYPE_TIMESTAMP
VARBINARY	SQL_BINARY

[1] This data type is a fixed-point number type for which precision and scale cannot be specified. By default, this data type maps to NUMERIC. However, you may use the Integer Field Mapping on page 88 option to map this type and other such types to BIGINT.

Snowflake Data Type	ODBC Data Type
VARCHAR	SQL_VARCHAR
VARIANT	SQL_VARCHAR

Driver specifications

This section describes the general functionality supported by the driver.

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

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

  The driver supports only the following Level 2 functions:

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

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

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

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

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

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

Additional information

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

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

Troubleshooting

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

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

Contacting Technical Support

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

https://www.progress.com/support

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

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

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.
- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.

- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

March 2020, Release 8.0.0 for the Progress DataDirect for ODBC for Snowflake Driver, Version 0001

2

Tutorials

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

For details, see the following topics:

- The Example application
- Microsoft Excel (Windows only)

The Example application

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

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

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

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

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

Note:

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

To use the Example application:

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

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

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

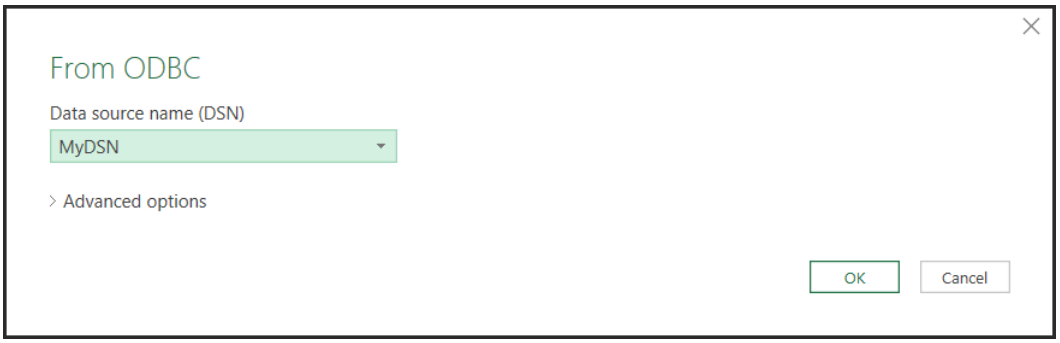
The results of your query are displayed. If example is unable to connect, the appropriate error message is returned.

Microsoft Excel (Windows only)

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

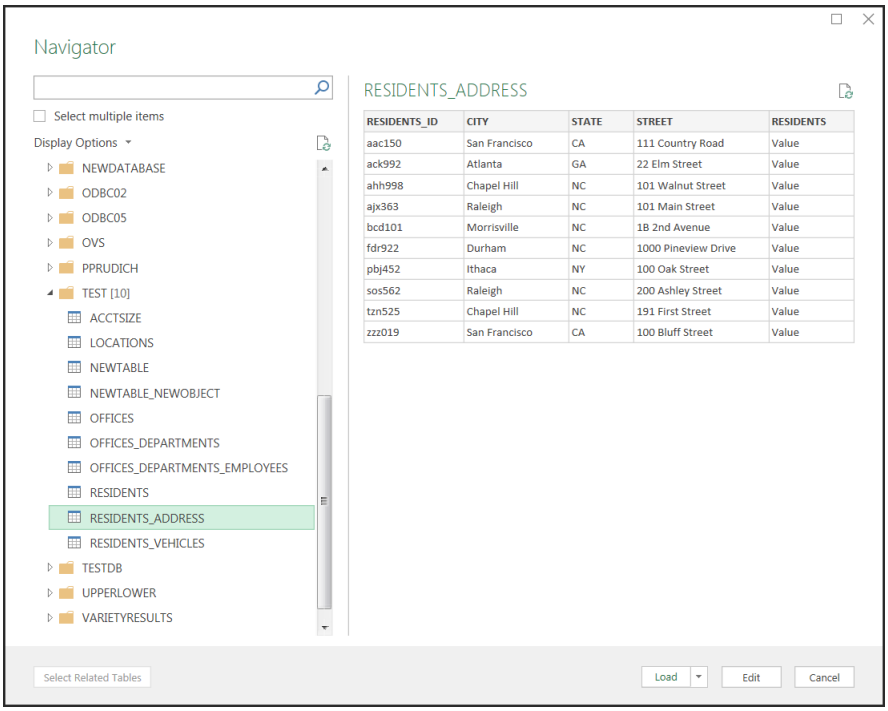
To use the driver to access data with Excel from the Data Connection Wizard:

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



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

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

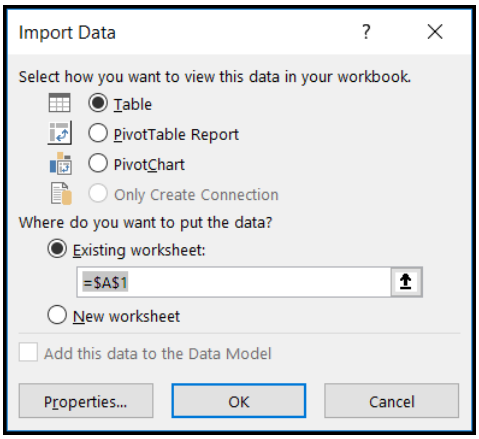


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

6. Load your data:

   - Click **Load** to import your data into your work sheet. Skip to the end.
   - Click **Load**>**Load To** to specify a location to import your data. Proceed to the next step.

7. The **Import Data** window appears.



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

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

3

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. Data sources store the Information that the driver needs to connect to a database. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on Linux), and file data sources. On Windows, user and system data sources are stored in the registry of the local computer. The difference is that only a specific user can access user data sources, whereas any user of the machine can access system data sources. On Windows and Linux, file data sources, which are simply text files, can be stored locally or on a network computer, and are accessible to other machines.

The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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

For details, see the following topics:

- Authentication
- Proxy server support
- Performance considerations

Environment settings

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

Windows environment variables

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

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

Linux environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

Executing these scripts sets the library search path environment variable, `LD_LIBRARY_PATH`.

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

```
env
```

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

ODBCINI

The product installer places a default system information file, named `odbc.ini`, that contains data sources in the product installation directory. See "Configuration through the system information (odbc.ini) file" for an explanation of the `odbc.ini` file. The system administrator can choose to rename the file and/or move it to another location. In either case, the environment variable `ODBCINI` must be set to point to the fully qualified path name of the `odbc.ini` file.

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

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

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

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

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

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

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

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

See also

Configuration through the system information (odbc.ini) file on page 36

ODBCINST

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

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

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

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

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

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

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

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

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

See also

DSN-less connections on page 39

DD_INSTALLDIR

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

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

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

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

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

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

1. The driver checks the `DD_INSTALLDIR` variable
2. The driver checks the `odbc.ini` or the `odbcinst.ini` files for the InstallDir keyword (see "Configuration through the system information (`odbc.ini`) file" for a description of the InstallDir keyword)

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

The next step is to test load the driver.

See also

Configuration through the system information (odbc.ini) file on page 36

The test loading tool

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

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

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

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

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

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

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

UTF-16 applications on Linux

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

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

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

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

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

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

Configuring data sources with the Configuration Manager

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

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

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

To configure and test a data source:

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

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

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

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

Note: Configuring data sources using the **File DSN** tab in the ODBC Administrator is not currently supported.

The Snowflake Configuration Manager window opens.

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

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

---

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

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

   c) Click **Execute**.

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

See also

Generating connection strings with the Configuration Manager on page 34
Connection option descriptions on page 73
Connection string examples on page 18

Generating connection strings with the Configuration Manager

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

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

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

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

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

See also

Configuring data sources with the Configuration Manager on page 33
Testing connections and queries with the Configuration Manager on page 41

Using a connection string

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

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

The DSN-less connection string has the form:

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

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

An example of a DSN connection string with overriding attribute values for the driver for Linux or Windows is:

```
DSN=Snowflake;USER=JohnQPublic;PWD=SECRET
```

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

```
FILEDSN=Snowflake.dsn;USER=JohnQPublic;PWD=SECRET
```

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

```
DRIVER=DataDirect 8.0 Snowflake;AN=MyAccountName.us-east-1;DB=MyDB;WH=MyWH;SCH=MySchema;
       USER=JohnQPublic;PWD=SECRET
```

See also

Connection option descriptions on page 73
Connection string examples on page 18

Additional configuration methods for Linux

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

Configuration through the system information (odbc.ini) file

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

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

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

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

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

```
Snowflake=DataDirect 8.0 Snowflake
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 73
File data sources on page 40
DSN-less connections on page 39

Sample default odbc.ini file

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

```
[ODBC Data Sources]
Snowflake=DataDirect 8.0 Snowflake

[Snowflake]
Driver=ODBCHOME/lib/ivsnowflake28.so
Description=DataDirect 8.0 Snowflake
AccessToken=
AccountName=
ArrowFallbackToJson=0
AuthenticationMethod=0
AuthURI=
```

```
ClientID=
ClientSecret=
ClientSessionKeepAlive=0
DatabaseName=
DataSourceName=
DisableSocksProxy=0
ExtendedOptions=
IntegerFieldMapping=1
JVMArgs=-Xmx1024m
JVMClasspath=
JVMPath=
LogConfigFile=
LoginTimeout=60
OAuthCode=
PartnerApplicationName=
Password=
ProxyHost=
ProxyPassword=
ProxyPort=
ProxyUser=
QueryTimeout=0
RedirectURI=
RefreshToken=
ReportCodepageConversionErrors=0
RoleName=
Schema=
Scope=
ServerPortNumber=19947
SQLEngineMode=0
TokenURI=
User=
UseSessionDatabaseForMetadata=0
Warehouse=HDP

[ODBC]
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10

[ODBC File DSN]
DefaultDSNDir=
UseCursorLib=0
```

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

- **To modify a data source:**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 73

DSN-less connections

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

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

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

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

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

```
DataDirect 8.0 Snowflake=Installed
```

This section also includes additional information for each driver.

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

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

See also

ODBCINST on page 31
Configuration through the system information (odbc.ini) file on page 36

Sample odbcinst.ini file

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

```
[ODBC Drivers]
DataDirect 8.0 Snowflake=Installed

[DataDirect 8.0 Snowflake]
Driver=ODBCHOME/lib/ivsnowflake28.so
JarFile=ODBCHOME/java/lib/snowflake.jar
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
SQLLevel=0
UsageCount=1

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

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

File data sources

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

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

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

```
[ODBC]
Driver=DataDirect 8.0 Snowflake
...
AccountName=MyAccountName.us-east-1
...
DatabaseName=MyDB
...
RoleName=MyRole
...
Schema=MySchema
...
Warehouse=MyWH
```

```
...
User=jsmith
...
Password=secret
```

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

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

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

or, on Linux:

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

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

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

```
FILEDSN=/home/users/john/filedsn/Snowflake.dsn;User=jsmith;PWD=Secret
```

See also

Configuring and connecting to data sources on page 29
DSN-less connections on page 39
Configuration through the system information (odbc.ini) file on page 36

Testing connections and queries with the Configuration Manager

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

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

To test your connection and query:

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

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

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

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

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

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

6. Click **Execute**.

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

See also

Configuring data sources with the Configuration Manager on page 33

Password Encryption Tool (UNIX/Linux only)

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

- Password
- Token

To use the Password Encryption Tool:

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

   `ddencpwd` *`password`*

   where:

   `password`

   is the password you want to encrypt.

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

   ```
   KeyPassword=returned_value
   ```

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

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

Using a logon dialog box

Some ODBC applications display a logon dialog box when you are connecting to a data source. The fields exposed in the logon dialog depend on the setting of the Authentication Method option. To connect, provide the values described in the following sections; then, click **OK** to complete the logon.

User ID and password authentication

In the dialog box, provide the following information:

- In the User Name field, type your logon ID.
- In the Password field, type your password.

Key-pair authentication

In the dialog box, provide the following information:

- In the Private Key Content field, specify the content of the private key you want to use for authentication.
- In the Private Key File field, specify the absolute path to the private key file you want to use for authentication.
- (optional) In the Private Key Passphrase field, specify the password for decrypting the private key or private key content you are using. Specify a value for this field If you are using an encrypted private key or private key content

OAuth authentication

In the dialog box, enter values for the OAuth 2.0 access flow or grant you are using. See "OAuth 2.0 configuration" for details.

Authentication

The Snowflake driver supports the following types of authentication:

- *User ID and password authentication* authenticates the user to the database using an user name and password.
- *Browser-based SSO authentication* authenticates the user to the database using a web browser in Microsoft Windows.
- *Key-pair authentication* authenticates the user using a pair of private and public keys.
- *OAuth 2.0 authentication* authenticates the user to the database using OAuth 2.0.

User ID and password authentication

To configure the driver to use user ID and password authentication:

- Configure the minimum options required for a connection:
  - Set the Account Name (`AccountName`) to specify the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
  - Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
  - Set the Warehouse (`Warehouse`) option to specify the name of the virtual warehouse to use, or an empty string.
  - Optionally, set the Schema (`Schema`) option to specify the schema to use for the specified database.
- Set the Authentication Method (`AuthenticationMethod`) option to `0` (user ID and password), which is the default.
- Set the User Name (`User`) option to specify your logon ID.
- Set the Password (`Password`) option to specify your password.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

The following examples demonstrate the connection information required to establish a session using user ID and password authentication.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;DatabaseName=MyDB;
       Schema=MySchema;Warehouse=MyWH;User=JohnQPublic;Password=SECRET
```

odbc.ini file (32-bit driver)

```
Driver=ODBCHOME/lib/ivsnowflake28.so
Description=My Snowflake Data Source
...
AccountName=MyAccountName.us-east-1
...
DatabaseName=MyDB;
...
Password=SECRET
...
Schema=MySchema
...
User=JohnQPublic
...
Warehouse=MyWH
...
```

See also

Connection option descriptions on page 73

Browser-based SSO authentication

Browser-based SSO authentication authenticates the user to the database using a web browser in Microsoft Windows. When using this method, the driver uses a web browser, ADFS, or any other SAML 2.0-compliant identify provider (IdP) when establishing a connection.

Important: Note that browser-based SSO is supported only when the SQL engine runs in direct mode or when the SQL engine server is launched from a command line. Browser-based SSO is not supported when running the SQL engine server as a Windows service.

To configure the driver to use browser-based SSO authentication:

- Configure the minimum options required for a connection:
  - Set the Account Name (`AccountName`) to specify the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
  - Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
  - Set the Warehouse (`Warehouse`) option to specify the name of the virtual warehouse to use, or an empty string.
  - Optionally, set the Schema (Schema) option to specify the schema to use for the specified database.
- Set the Authentication Method (`AuthenticationMethod`) option to `45`.
- Set the User Name (`User`) option to specify your logon ID.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

The following examples demonstrate the connection information required to establish a session using user ID and password authentication.

Connection string

```
DRIVER=DataDirect 8.0
Snowflake;AccountName=MyAccountName.us-east-1;AuthenticationMethod=45;
       DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;User=JohnQPublic;
```

odbc.ini file (32-bit driver)

```
Driver=ODBCHOME/lib/ivsnowflake28.so
Description=My Snowflake Data Source
...
AccountName=MyAccountName.us-east-1
...
AuthenticationMethod=45
...
DatabaseName=MyDB
...
Schema=MySchema
...
User=JohnQPublic
...
Warehouse=MyWH
...
```

See also

Connection option descriptions on page 73

Key-pair authentication

The driver supports key-pair authentication. Key-pair authentication allows you to authenticate to Snowflake using a pair of private and public keys.

Note: The keys used for authentication must be RSA keys that are at least 2048 bits long.

To configure the driver to use browser-based SSO authentication:

- Configure the minimum options required for a connection:
  - Set the Account Name (`AccountName`) to specify the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
  - Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
  - Set the Warehouse (`Warehouse`) option to specify the name of the virtual warehouse to use, or an empty string.
  - Optionally, set the Schema (Schema) option to specify the schema to use for the specified database.
- Set the Authentication Method (`AuthenticationMethod`) option to `21`.
- Set the User Name (`User`) option to specify your logon ID.
- Configure one of the following connection options to specify the private key:
  - Set the Private Key File (`PrivateKeyFile`) option to specify the absolute path to the private key file you want to use for authentication.
  - Set the Private Key Content (`PrivateKeyContent`) option to specify the content of the private key you want to use for authentication.
- If you are using an encrypted private key or private key content, set the Private Key Passphrase (`PrivateKeyPassphrase`) option to specify the password for decrypting the private key or private key content you are using.
- Optionally, specify values for any additional options you want to configure. See "Connection option descriptions" for a complete list of options.

Note: If the encryption schema you are using to generate the encrypted private keys is not compatible with the native encryption libraries of your JRE, the JRE will return an error. To resolve this issue, either generate encrypted private keys using an encryption schema that is compatible with your JRE or add a third-party Java cryptography library to your application (for example, Bouncy Castle) that supports the encryption schema you are using. For example, to add Bouncy Castle to your application, add the following line to the `java.security` file: `security.provider.n=org.bouncycastle.jce.provider.BouncyCastleProvider`, and then add the Bouncy Castle jars to the classpath of your JRE.

The following examples demonstrate the connection information required to establish a session using key-pair authentication.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   AuthenticationMethod=21;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;
   User=JohnQPublic;PrivateKeyFile=C:\Program Files\privatekey.p8;
   PrivateKeyPassphrase=abc123;
```

odbc.ini file (32-bit driver)

```
Driver=ODBCHOME/lib/ivsnowflake28.so
Description=My Snowflake Data Source
...
AccountName=MyAccountName.us-east-1
...
AuthenticationMethod=21
...
User=JohnQPublic
...
DatabaseName=MyDB
...
PrivateKeyFile=C:\Program Files\privatekey.p8
...
PrivateKeyPassphrase=abc123
...
Schema=MySchema
...
Warehouse=MyWH
...
```

See also

Connection option descriptions on page 73

OAuth 2.0 authentication

The driver may be used to access Snowflake resources with Snowflake OAuth, Snowflake's built-in OAuth service. The driver supports the following OAuth 2.0 flows and grants: the access token flow, the authorization code grant, the client credentials grant, and the refresh token grant.

The following workflow describes the process for setting up OAuth 2.0 access.

1. Create a Snowflake OAuth security integration. You must create a security integration to set up OAuth in Snowflake. You can then obtain client information, such as client ID, client secret, authorization URI, redirect URI, and token URI.
2. Obtain access tokens with the Configuration Manager. Before you can configure the driver, you must obtain any required access token.
3. Configure the driver to use OAuth 2.0. You can configure the driver to access Snowflake resources using the access token flow, authorization code grant, client credentials grant, and refresh token grant.

Creating an OAuth security integration and obtaining client information

An OAuth security integration is a Snowflake object that enables client application access to Snowflake resources using OAuth 2.0. There are two types of OAuth security integrations in Snowflake: Snowflake OAuth and External OAuth. Snowflake OAuth is Snowflake's built-in OAuth service, while External OAuth allows you to use an external authorization service. It is during the process of creating an OAuth integration that you obtain the client information the driver requires to connect with Snowflake.

See the following topics for details.

- Creating a Snowflake OAuth security integration
- Creating an External OAuth security integration

Note: For additional details, refer to OAuth in the Snowflake documentation.

Creating a Snowflake OAuth security integration

Snowflake OAuth is Snowflake's built-in OAuth service. When you create a Snowflake OAuth security integration, you effectively register your client application with Snowflake. After you create the security integration, you can obtain the client information (client ID, client secret, authorization endpoint, and token endpoint) required for configuring the driver.

Take the following steps to create a Snowflake security integration and obtain required client information.

Note: The Snowflake user must have either the ACCOUNTADMIN role or the global CREATE INTEGRATION privilege to execute the `create security integration` command.

1. Log in to Snowflake.
2. Open **Worksheets** or select the scheme you are working with.

   Note: Administrators may check to see whether the SYSADMIN role has the required privileges for the warehouse by navigating to **Admin > Warehouses > *warehouse_name* > Edit > Privileges**.

3. Run the following command to create the security integration.

   Note: For details on parameters, refer to CREATE SECURITY INTEGRATION (Snowflake OAuth) in the Snowflake documentation.

   ```
   create security integration integration_name
    type = oauth
    enabled = true
    oauth_client = custom
    oauth_client_type = confidential
    oauth_redirect_uri =redirect_uri
    oauth_issue_refresh_tokens = true
    oauth_refresh_token_validity=7776000;
   ```

   where:

   integration_name

   is the name of the security integration.

   redirect_uri

   is the client URI. The web browser is redirected to this URI after authorization. For example, to test, you might use `http://localhost`.

4. Run the following `describe` command to obtain the client ID, authorization URI, and token URI.

   ```
   describe security integration integration_name;
   ```

5. Run the following `select` command to obtain the client secret.

   ```
   select SYSTEM$SHOW_OAUTH_CLIENT_SECRETS(integration_name);
   ```

   Note: Two client secrets will be returned. Either may be used to configure the driver.

Results

You have created a Snowflake OAuth security integration. Next you must obtain an access token. See "Obtaining access tokens using the Configuration Manager" for more information.

See also

Obtaining access and refresh tokens using the Configuration Manager on page 53

Creating an External OAuth security integration

External OAuth allows you to use an external authorization service to access Snowflake. To implement OAuth for a client application, you must configure the external authorization service as well as Snowflake. During this process, you obtain the client information (client ID, client secret, authorization endpoint, and token endpoint) required to configure the driver.

Note: The following instructions provide guidance on developing an External OAuth integration with Okta. However, Snowflake supports a number of other external authorization services. Refer to External OAuth in the Snowflake documentation for details.

The following workflow describes the process for creating an External OAuth security integration with Okta.

1. Okta: Create an OAuth client on page 50
2. Okta: Create the OAuth authorization server on page 51
3. Snowflake: Create External OAuth security integration on page 51

Okta: Create an OAuth client

1. Log in to Okta, and navigate to the Admin Console.
2. Navigate to **Applications > Applications**. Then, click **Create App Integration**.
3. For **Sign-in method**, select **OIDC**.
4. For **Application type**, select **Native Application**.
5. Click **Next**.
6. On the **App Integration** screen, enter the **App integration name**, and select the grant types you plan to use.
7. For **Sign-in redirect URIs**, enter your Snowflake account URL.

   Note: For example, to test, you might use `http://localhost`.

   ---

8. Click **Save**.
9. From the **General** tab of your integration, click **Edit** next to **Client Credentials**.
10. Select **Use Client Authentication**.
11. For **Client authentication**, select **Client secret**.
12. Click **Save**.
13. From the **General** tab of your integration, save the **Client ID** and **Client Secret** values.
14. Select the **Assignments** tab, and assign the current user to the app.

Result: An Okta OAuth client has been created. In addition, you have saved the client ID and client secret which will be needed to configure the driver.

Okta: Create the OAuth authorization server

1. From the Okta Admin Console, navigate to **Security > API** and click **Add Authorization Server**.
2. For **Audience**, enter the Snowflake account URL. Then, click **Save**.
3. From the **Settings** tab, copy and save the **Issuer** value.
4. From the **Settings** tab, open the **Metadata URI** and save the following values.
   - **JWS keys URL:** the value of the `jwks_uri` parameter
   - **Authorization URL:** the value of the `authorization_endpoint` parameter
   - **Token URL:** the value of the `token_endpoint` parameter
5. From the **Scopes** tab, click **Add Scope**.
6. Enter the Snowflake scope and scope information.

   Note: Snowflake scopes are typically tied to Snowflake roles. For example: `session:role:SYSADMIN`.

7. Click **Save**.
8. From the **Access Policies** tab, click **Add New Access Policy**.
9. Select **The following clients**, and add the OAuth client application create in "Okta: Create an OAuth client".
10. Click **Create Policy**.
11. Click **Add Rule**, and make selections based on your requirements.

    Note:
    - **Client Credentials** and **Resource Owner Password** are required for Snowflake.
    - **Authorization Code** should be selected if you plan to use the authorization code grant or use the driver Configuration Manager to generate access and refresh tokens.
12. Click **Create rule**.

Result: An Okta authorization server has been created to use with an External OAuth security integration in Snowflake. In addition, you have recorded the token URI and the authorization URI which are required to configure the driver for some OAuth 2.0 grant types.

Snowflake: Create External OAuth security integration

Note: The Snowflake user must have either the ACCOUNTADMIN role or the global CREATE INTEGRATION privilege to execute the `create security integration` command.

1. Log in to Snowflake.
2. Open **Worksheets** or select the schema you are working with.

   Note: Administrators may check to see whether the SYSADMIN role has the required privileges for the warehouse by navigating to **Admin > Warehouses > *`warehouse_name`* > Edit > Privileges**.

3. Run the following command to create the security integration.

Note: For more details on parameters, refer to CREATE SECURITY INTEGRATION (External OAuth) in the Snowflake documentation.

```
create security integration integration_name
    type = external_oauth
    enabled = true
    external_oauth_any_role_mode = 'ENABLE'
    external_oauth_type = okta
    external_oauth_issuer = 'okta_issuer'
    external_oauth_jws_keys_url = 'okta_jws_keys_url'
    external_oauth_audience_list = ('audience_list')
    external_oauth_token_user_mapping_claim = 'sub'
    external_oauth_snowflake_user_mapping_attribute = 'login_name';
```

where:

integration_name

is the name of the security integration.

okta_issuer

is the URL that defines the Okta OAuth 2.0 authorization server. Obtained in Step 3 of "Okta: Create the OAuth authorization server". For example: `https://dev-123456.okta.com/oauth2/abcdefg`.

okta_jws_keys_url

is the Okta URL where public keys may be downloaded to validate an External OAuth access token. Obtained in Step 4 of "Okta: Create the OAuth authorization server". For example: `https://dev-123456.okta.com/oauth2/abcdefg/v1/keys`.

audience_list

is your Snowflake Account URL. For example: `https://myorg-account.us-east-1.snowflakecomputing.com`.

`external_oauth_token_user_mapping_claim`

specifies a key that indicates the user associated with the access token. Typically, in Okta, the `sub` key passes the Okta user associated with a given access token.

`external_oauth_snowflake_user_mapping_attribute`

specifies the Snowflake user attribute that should be used to map the access token to a Snowflake user record. In the example, the Snowflake user `login_name` attribute is being used for this purpose. Therefore, the value of the `sub` key must match the value of `login_name` to complete the OAuth flow and grant the client application access to Snowflake resources.

What's next

If you are using the client credentials grant or the authorization code grant, proceed to the corresponding topic for guidance on configuring the driver.

- Authorization code grant
- Client credentials grant

If you are using the access token flow or refresh token grant, see Obtaining access and refresh tokens using the Configuration Manager.

Obtaining access and refresh tokens using the Configuration Manager

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

You need the following information before you begin.

- Authorization URI: The endpoint for obtaining an authorization code from a third-party authorization service
- Token URI: The endpoint used to exchange authentication credentials for access tokens
- Redirect URI: The endpoint that the client is returned to after authenticating with the service
- Client ID: The client ID for your application
- Client Secret: The client secret for your application

The following steps describe how you can use the Progress DataDirect Snowflake Configuration Manager to obtain access and refresh tokens for either the access token flow or the refresh token grant. In addition, the Configuration Manager produces a connection string that you can use in your application.

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

1. Open the Windows ODBC Administrator.
2. Open the Configuration Manager through the **User DSN** or **System DSN** tab.
   - **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the Configuration Manager.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the Configuration Manager in your browser.

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

   Note: Configuring data sources using the **File DSN** tab in the ODBC Administrator is not currently supported.

3. On the **Connection** tab, provide values for the following fields:

   For user ID and password authentication:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Account Name**: Type the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
   - **Database Name**: Type the name of database to which you want to connect.
   - **Password**: Type the password that is used to connect to the Snowflake instance.

- **Schema**: Type the schema for the specified database.
- **User**: Type the user name that is used to connect to the Snowflake instance.
- **Warehouse**: Type the name of the virtual warehouse.
- **Role Name**: Type the role for access control in the Snowflake session initiated by the driver.

Note: User and Password connection options can also be specified in the logon dialog box or passed by your application.

Note: The driver supports a number of authentication methods. See Authentication on page 43 for more information.

4. Set **Authentication Method** to `24 - OAuth2`.
5. Provide the following information in the fields provided.
   - **Authorization URI**
   - **Token URI**
   - **Redirect URI**
   - **Client ID**
   - **Client Secret**
6. Retrieve the access and refresh tokens.
   a) Click **Fetch OAuth Token**.
   b) If the logon popup appears, enter your credentials. (This popup may not appear if you previously logged on.)
   c) If the consent popup appears, provide consent, allowing the Configuration Manager to retrieve the tokens. (This popup may not appear if you previously provided consent to the Configuration Manager.)
   d) The **Access Token** and **Refresh Token** fields populate with values retrieved from the OAuth authorization server.
7. Click **Test Connect** to verify connectivity and run SQL queries against the service.

Results:

The **Access Token** and **Refresh Token** fields include access and refresh tokens that you can use to implement OAuth 2.0.

The connection string in the **Connection String** field may be copied and used in your ODBC application to connect with your service.

Note:

Not all the values in the resulting connection string are required. However, the connection string can be copied directly to a location that can be used by your application. The driver ignores any values that do not apply to your OAuth implementation.

For example, the connection string derived from the Configuration Manager might include the following options.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=account_name.us-east-1;
       DatabaseName=MyDB;Schema=MySchema;Warehouse=accounting;AuthenticationMethod=24;
       AuthURI=auth_uri;TokenURI=token_uri;ClientID=client_id;
```

```
        ClientSecret=client_secret;AccessToken=access_token;
        RefreshToken=refresh_token;
```

However, only the following options are required for a refresh token grant connection string.

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=account_name.us-east-1;DatabaseName=MyDB;
      Schema=MySchema;Warehouse=accounting;AuthenticationMethod=24;TokenURI=token_uri;

        RedirectURI=redirect_uri;ClientID=client_id;
        ClientSecret=client_secret;RefreshToken=refresh_token;
```

See also

Configuring data sources with the Configuration Manager on page 33

Configuring OAuth 2.0 authentication

The driver supports the following OAuth 2.0 flow and grant types.

- Access token flow on page 55
- Authorization code grant on page 56
- Client credentials grant
- Refresh token grant on page 59

Access token flow

Note: For OAuth 2.0 authentication, using a refresh token is more user-friendly and secure than using an access token. The access token expires every 10 minutes and may need to be generated multiple times a day, which can prolong the process of establishing a connection and cause a security risk.

The access token authentication flow passes the token directly from the client to the Snowflake instance for authentication. The token is obtained from sources external to the flow and specified using the Access Token (`AccessToken`) option.

To configure the driver to use the access token flow for OAuth 2.0 authentication:

- Set the Authentication Method (`AuthenticationMethod`) option to `24`.
- Set the Account Name (`AccountName`) option to specify the name of your account and the region where it is hosted. For example, *`account_name`*`.us-east-1`.
- Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
- Set the Schema (`Schema`) option to specify the default schema to use for the specified database once connected. The specified schema should be an existing schema for which the specified default role has privileges.
- Set the Warehouse (`Warehouse`) option to specify the virtual warehouse to use once connected. The specified warehouse should be an existing warehouse for which the specified default role has privileges.
- Set the Access Token (`AccessToken`) option to the value of the token obtained from external sources.

  Important:

  - The access token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

- Access tokens expire ten minutes after generation. Once connected, the access token remains valid till the session is disconnected.

The following examples show the connection information required to establish a session using the access token flow.

Connection string

```
DRIVER=DataDirect 8.0
Snowflake;AuthenticationMethod=24;AccountName=account_name.us-east-1;
  DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;Token=abc12cd34efg5678h9ij87klm6543no;
```

odbc.ini file (32-bit)

```
Driver=ODBCHOME/lib/ivsnowflake28.yy
...
AuthenticationMethod=24
...
AccountName=account_name.us-east-1
...
DatabaseName=MyDB
...
Schema=MySchema
...
Token=abc12cd34efg5678h9ij87klm6543no
...
Warehouse=MyWH
...
```

See also

Creating a Snowflake OAuth security integration on page 49
Obtaining access and refresh tokens using the Configuration Manager on page 53
Connection option descriptions on page 73

Authorization code grant

The authorization code grant is a commonly used authentication flow for web and native applications. It provides secure connections by requiring multiple points of authentication before permitting access to data. When using the authorization code flow, the application is first redirected to the location hosting the temporary authorization code and retrieves it. Next, after being redirected to the location specified by the Redirect URI (`RedirectURI`) option, the application exchanges the authorization code, client ID, and client secret for the access token.

To use an authorization code grant:

- Set the Authentication Method (`AuthenticationMethod`) option to `24`.
- Set the Account Name (`AccountName`) option to specify the name of your account and the region where it is hosted. For example, *`account_name`*`.us-east-1`.
- Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
- Set the Schema (`Schema`) option to specify the default schema to use for the specified database once connected. The specified schema should be an existing schema for which the specified default role has privileges.
- Set the Warehouse (`Warehouse`) option to specify the virtual warehouse to use once connected. The specified warehouse should be an existing warehouse for which the specified default role has privileges.
- Set the Authorization URI (`AuthURI`) option to the endpoint used to obtain the authorization code from the authorization service.

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

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

- Set the Redirect URI (`RedirectURI`) option to specify the endpoint that the client is returned to after authenticating with the service. This value must match the redirect URI specified in the Snowflake OAuth security integration.

The following examples show the connection information required to establish a session using the authorization code grant.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AuthenticationMethod=24;
  AccountName=account_name.us-east-1;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;

  AuthURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/authorize;
  ClientId=cd34efg5678h9ij87klm6543no32pqr10st987;ClientSecret=098zyx765wvu432tsr123qpo456;

  RedirectUri=https://lvn.me/app_callback.html;
```

odbc.ini file (32-bit)

```
Driver=ODBCHOME/lib/ivsnowflake28.yy
...
AuthenticationMethod=24
...
AccountName=account_name.us-east-1
...
DatabaseName=MyDB
...
Schema=MySchema
...
Warehouse=MyWH
...
AuthURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/authorize
...
ClientId=cd34efg5678h9ij87klm6543no32pqr10st987
...
ClientSecret=098zyx765wvu432tsr123qpo456
...
RedirectUri=https://lvn.me/app_callback.html
...
```

See also

Creating a Snowflake OAuth security integration on page 49
Connection option descriptions on page 73

Client credentials grant

The authentication flow for the client credentials grant exchanges client credentials for the access token at the location specified by the Toke URI (`TokenURI`) option. Web-based login and consent are not required.

To configure the driver to use a client credentials grant:

- Set the Authentication Method (`AuthenticationMethod`) option to `24`.

- Set the Account Name (`AccountName`) option to specify the name of your account and the region where it is hosted. For example, *`account_name`*`.us-east-1`.
- Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
- Set the Schema (`Schema`) option to specify the default schema to use for the specified database once connected. The specified schema should be an existing schema for which the specified default role has privileges.
- Set the Warehouse (`Warehouse`) option to specify the virtual warehouse to use once connected. The specified warehouse should be an existing warehouse for which the specified default role has privileges.
- Set the Client ID (`ClientID`) option to specify the client ID for your application.
- Set the Client Secret (`ClientSecret`) option to specify 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 Token URI (`TokenURI`) option to specify the endpoint from which the driver fetches access tokens.

The following examples show the connection information required to establish a session using the client credentials grant.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AuthenticationMethod=24;
  AccountName=account_name.us-east-1;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;

  ClientId=cd34efg5678h9ij87klm6543no32pqr10st987;ClientSecret=098zyx765wvu432tsr123qpo456;

  TokenURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/token-request;
```

odbc.ini file (32-bit)

```
Driver=ODBCHOME/lib/ivsnowflake28.yy
...
AuthenticationMethod=24
...
AccountName=account_name.us-east-1
...
DatabaseName=MyDB
...
Schema=MySchema
...
Warehouse=MyWH
...
ClientId=cd34efg5678h9ij87klm6543no32pqr10st987
...
ClientSecret=098zyx765wvu432tsr123qpo456
...
TokenURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/token-request
...
```

See also

Creating a Snowflake OAuth security integration on page 49
Connection option descriptions on page 73

Refresh token grant

The refresh token grant is used to request a new access token or renew an expired one by exchanging the refresh token at the endpoint specified by the Token URI (`TokenURI`) option.

To configure the driver to use the refresh token grant for OAuth 2.0 authentication:

- Set the Authentication Method (`AuthenticationMethod`) option to `24`.
- Set the Account Name (`AccountName`) option to specify the name of your account and the region where it is hosted. For example, *`account_name`*`.us-east-1`.
- Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
- Set the Schema (`Schema`) option to specify the default schema to use for the specified database once connected. The specified schema should be an existing schema for which the specified default role has privileges.
- Set the Warehouse (`Warehouse`) option to specify the virtual warehouse to use once connected. The specified warehouse should be an existing warehouse for which the specified default role has privileges.
- Set the Client ID (`ClientID`) option to specify the client ID for your application.
- Set the Client Secret (`ClientSecret`) option to specify 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 Token URI (`TokenURI`) option to specify the endpoint from which the driver fetches access tokens.
- Set the Refresh Token (`RefreshToken`) option to specify the refresh token used to request a new access token or renew an expired one.

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

The following examples show the connection information required to establish a session using the refresh token grant.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AuthenticationMethod=24;
  AccountName=account_name.us-east-1;DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;

  ClientId=cd34efg5678h9ij87klm6543no32pqr10st987;ClientSecret=098zyx765wvu432tsr123qpo456;

   TokenURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/token-request;
   RefreshToken=abc12cd34efg5678h9ij87klm6543no32pqr10;
```

odbc.ini file (32-bit)

```
Driver=ODBCHOME/lib/ivsnowflake28.yy
...
AuthenticationMethod=24
...
AccountName=account_name.us-east-1
...
DatabaseName=MyDB
...
```

```
Schema=MySchema
...
Warehouse=MyWH
...
ClientId=cd34efg5678h9ij87klm6543no32pqr10st987
...
ClientSecret=098zyx765wvu432tsr123qpo456
...
TokenURI=https://account_name.us-east-1.snowflakecomputing.com/oauth/token-request
...
RefreshToken=abc12cd34efg5678h9ij87klm6543no32pqr10
...
```

See also

Creating a Snowflake OAuth security integration on page 49
Obtaining access and refresh tokens using the Configuration Manager on page 53
Connection option descriptions on page 73

Proxy server support

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 greenplum.jar com.acme.myapp.Main
```

Alternatively, you can use the Proxy Host (`ProxyHost`), Proxy Port (`ProxyPort`), Proxy User (`ProxyUser`), and Proxy Password (`ProxyPassword`) connection options. See "Connection Option Descriptions" for details about these attributes.

To configure the driver to connect through a proxy server with the user ID and password authentication:

- Configure the minimum options required for a connection:
  - Set the Account Name (`AccountName`) to specify the name of your account and the region where it is hosted. For example, `my_account.us-east-1`.
  - Set the Database Name (`DatabaseName`) option to specify the name of database to which you want to connect.
  - Set the Warehouse (`Warehouse`) option to specify the name of the virtual warehouse to use, or an empty string.
  - Optionally, set the Schema (`Schema`) option to specify the schema to use for the specified database.
  - Configure your authentication related options. See "Authentication" for more information on these options.

- Set the Proxy Host (`ProxyHost`) option to specify the proxy server to use for the first connection.
- Set the Proxy Password (`ProxyPassword`) option to specify the password needed to connect to a proxy server for the first connection.
- Set the Proxy Port (`ProxyPort`) option to specify the port number where the proxy server is listening for requests for the first connection. The default is `0`.
- Set the Proxy User (`ProxyUser`) option to specify the user name needed to connect to a proxy server for the first connection.

The following examples include the options required for using a proxy server with user ID and password authentication.

Connection string

```
DRIVER=DataDirect 8.0 Snowflake;AccountName=MyAccountName.us-east-1;
   DatabaseName=MyDB;Schema=MySchema;Warehouse=MyWH;User=JohnQPublic;
   Password=SECRET;ProxyHost=pserver;ProxyPassword=proxys3cr3t;
   ProxyPort=1234;ProxyUser=jsmith;
```

odbc.ini

```
[Snowflake]
Driver=ODBCHOME/lib/xxsnowflake28.yy
...
AccountName=MyAccountName.us-east-1
...
DatabaseName=MyDB
...
HostName=myserver
...
Password=secret
...
ProxyHost=pserver
...
ProxyPassword=proxys3cr3t
...
ProxyPort=1234
...
ProxyUser=jsmith
...
Schema=MySchema
...
User=JohnQPublic
...
Warehouse=MyWH
...
```

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

See also

Connection option descriptions on page 73
Authentication on page 43
Using a connection string on page 35
Connection option descriptions on page 73

Performance considerations

Arrow Fallback To JSON (ArrowFallbackToJson): Arrow Fallback To JSON can be configured to allow the driver to fall the JSON query format when the arrow format is not properly initialized (`ArrowFallbackToJson=0 | 1`). By falling back to JSON query format, the driver can continue to execute queries that would normally return an exception; however, there is a significant performance tradeoff from not using the high-speed arrow transfer. For ideal performance, configure your JVM properly by specifying the `--add-opens=java.base/java.nio=ALL-UNNAMED` option value and disable this connection option (`ArrowFallbackToJson= 2`).

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

4

Using the SQL engine server

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

The driver supports the following SQL engine modes:

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

By default:

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

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

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

For details, see the following topics:

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

Configuring server mode using the Configuration Manager

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

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

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

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

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

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

     32-bit: `19948`

     64-bit: `19947`
   - **JVM Classpath**: Specifies the CLASSPATH for the JVM used by the driver. See "JVM Classpath" for details. The default depends on your platform:

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

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

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

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

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

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

See also

Configuring the SQL engine server using Java options on page 65
JVM Classpath on page 89
JVM Arguments on page 88

Stopping the SQL engine server using the Configuration Manager

To stop the SQL engine server:

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

Configuring the SQL engine server using Java options

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

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

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

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

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

For example:

```
java -Xmx1024m -cp "/opt/Progress/DataDirect/ODBC_80_64bit/java/lib/snowflake.jar"
        com.ddtek.phoenix.sql.Server -port 19947 -Dhttp.proxyHost=myhost@mydomain.com
        -Dhttp.proxPort=12345 -Dhttp.proxyUser=JohnQPublic -Dhttp.proxyPassword=secret
```

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

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

Table 2: SQL engine server Java options

Java Option	Description
Required Java Options	
`-cp`	Specifies the CLASSPATH for the Java Virtual Machine (JVM) used by the driver. The CLASSPATH is the search string the JVM uses to locate the Java jar files the driver needs. The driver's JVM is located on the following path: *`install_dir`*`/java/lib/snowflake.jar`
`-port`	Specifies a valid port on which the SQL engine listens for requests from the driver. We recommend specifying one of the following values: • `19948` (32-bit drivers) • `19947` (64-bit drivers)
Optional Java Options	
`-Xmx`	Specifies the maximum memory heap size, in megabytes, for the JVM. The default size is determined by your JVM. We recommend specifying a size no smaller than `1024`. **Note:** Although this option is not required to start the SQL engine server, we highly recommend specifying a value.
`-Dhttp.proxyHost`	Specifies the host name of the proxy server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
`-Dhttp.proxyPort`	Specifies the port number where the proxy server is listening for HTTP and/or HTTPS requests.
`-Dhttp.proxyUser`	Specifies the user name needed to connect to the proxy server.
`-Dhttp.proxyPassword`	Specifies the password needed to connect to the proxy server.

Stopping the SQL engine server

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

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

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

Configuring Java logging for the SQL engine server

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

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

5

Additional features and functionality

The following section describes additionally supported features and functionality that are specific to the driver.

For details, see the following topics:

- Timeouts
- Using identifiers
- Using COPY command

Timeouts

The following type of timeout situation can occur when connecting to Snowflake:

- **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. The driver automatically attempts to re-establish a new session if the driver receives a session timeout error from a data source. The driver uses the initial servername, port (if appropriate), remote user ID, and remote password (encrypted) to re-establish the session. If the attempt fails, the driver returns an error indicating that the session timed out and the attempt to re-establish the session failed.

See also

Connection option descriptions on page 73

Using identifiers

Identifiers are used to refer to objects exposed by the driver, such as tables, columns, or caches. The driver supports both unquoted and quoted identifiers for naming objects. An unquoted identifier must start with an ASCII alpha character and can be followed by zero or more ASCII alpha or numeric characters.

Quoted identifiers must be enclosed in backticks (``). A quoted identifier can contain any Unicode character, including the space character, and is case-sensitive. The Snowflake driver recognizes the Unicode escape sequence \uxxxx as a Unicode character.

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

Using COPY command

The driver supports using the COPY command for loading and unloading data from local file systems and cloud platforms, such as Amazon S3, Google Cloud, and Microsoft Azure. The data is moved from staged files to an existing table and vice versa.

Snowflake supports the following stages for loading and unloading of files.

- **User**: This stage is used for files that are accessed by a single user and copied into multiple tables.
- **Table**: This stage is used for files that are accessed by multiple users and copied into a single table.
- **Internal Named**: This stage is used for named database objects that provide flexibility and security during data loading.

The driver supports all the copy options and format type options supported by Snowflake. The process of data loading and unloading depends on your environment. Refer to the following topics in your Snowflake documentation for further details.

- Data Loading

  https://docs.snowflake.com/en/user-guide-data-load.html

- Data Unloading

  https://docs.snowflake.com/en/user-guide-data-unload.html

Parameter metadata support

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

Insert and Update statements

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

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

where:

operator

is any of the following SQL operators:

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

.

Select statements

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

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

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

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

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

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

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

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

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

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

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

Joins are supported. For example:

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

Fully qualified names and aliases are supported. For example:

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

6

Connection option descriptions

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

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

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

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

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

General options

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

Table 3: General options

Attribute (Short Name)	Default
AccountName (an)	No default value
DatabaseName (db)	No default value

Attribute (Short Name)	Default
DataSourceName (dsn)	No default value
Description (desc)	No default value
RoleName (role)	No default value
Schema (schm)	No default value
Warehouse (wh)	No default value

User ID and password authentication options

The following table summarizes options used for user ID and password authentication.

Table 4: User ID and password options

Attribute (Short Name)	Default
AuthenticationMethod (am)	`0` (UserIDPassword)
Password (pwd)	No default value
User	No default value

Browser-based SSO authentication options

The following table summarizes options used for browser-based SSO authentication.

Table 5: Browser-based SSO authentication options

Attribute (Short Name)	Default
AuthenticationMethod (am)	`0` (UserIDPassword)
User	No default value

OAuth 2.0 options

The following table summarizes options used for OAuth 2.0 authentication.

Table 6: OAuth 2.0 options

Attribute (Short Name)	Default
AccessToken (atok)	No default value
AuthenticationMethod (am)	`0` (UserIDPassword)
AuthURI (o2au)	No default value
ClientID (cid)	No default value

Attribute (Short Name)	Default
ClientSecret (clse)	No default value
OAuthCode (oac)	No default value
RedirectURI (o2ru)	No default value
RefreshToken (rtok)	No default value
Scope (oas)	No default value
TokenURI (o2tu)	No default value

Key-pair authentication options

The following table summarizes options used for key-pair authentication.

Table 7: Key-pair options

Attribute (Short Name)	Default
AuthenticationMethod (am)	`0` (UserIDPassword)
PrivateKeyContent (pkc)	No default value
PrivateKeyFile (pkf)	No default value
PrivateKeyPassphrase (pkpp)	No default value
User	No default value

Proxy server options

The following table summarizes proxy server options.

Table 8: Proxy server options

Attribute (Short Name)	Default
ProxyHost (pxhn)	No default value
ProxyPassword (pxpw)	No default value
ProxyPort (pxpt)	No default value
ProxyUser (pxun)	No default value

SQL engine options

The following table summarizes SQL engine options.

Table 9: SQL engine attributes

Attribute (Short Name)	Default

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

JVMClassPath (jvmcp)	*`install_dir`*`\java\lib\snowflake.jar`
JVMPath (jvmp)	*`install_dir`*`\jre\bin\java.exe`

ServerPortNumber (sport)	For a 32-bit driver:
	`19948`
	For a 64-bit driver:
	`19947`

SQLEngineMode (sem)	For Windows: `0` (Auto) For UNIX/Linux: `2` (Direct)
SQLService (ss)	No default value

Timeout options

The following table summarizes timeout options.

Table 10: Timeout attributes

Attribute (Short Name)	Default
LoginTimeout (lt)	`60`
QueryTimeout (qt)	`0`

Additional attributes

The following table summarizes additional attributes.

Table 11: Additional attributes

Attribute (Short Name)	Default
ArrowFallbackToJson (aftj)	`0` (Enable)
ClientSessionKeepAlive (cska)	`0` (false)
DisableSocksProxy (dsp)	`0` (false)

Attribute (Short Name)	Default
ExtendedOptions (xo)	No default value
IntegerFieldMapping (ifm)	`1` (Numeric)
LogConfigFile (lgcf)	`ddlogging.properties`
PartnerApplicationName (pan)	No default value
ReportCodepageConversionErrors (rcce)	`0` (Ignore Errors)
UseSessionDatabaseForMetadata (usdm)	`0` (false)

For details, see the following topics:

- Access Token
- Account Name
- Arrow Fallback To JSON
- Authorization Code
- Authentication Method
- Authorization URI
- Client ID
- Client Secret
- Client Session Keep Alive
- Data Source Name
- Database Name
- Description
- Disable SOCKS Proxy
- Extended Options
- Integer Field Mapping
- JVM Arguments
- JVM Classpath
- JVM Path
- Log Config File
- Login Timeout
- Partner Application Name
- Password

Access Token

Attribute

AccessToken (atok)

Purpose

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

Valid Values

`String`

where:

String

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

Notes

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

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Account Name

Attribute

AccountName (acn)

Purpose

Specifies the full name of your account and the region where it is hosted. The full account name may include additional segments that denote region and cloud platform.

Valid Values

String

where:

String

is the full name of an account with region and cloud platform.

Example

Cloud Platform/region	Full Account Name
AWS	
US East (N.Virginia)	`account_name.us-east-1`
US East (Ohio)	`account_name.us-east-2.aws`
US West (Oregon)	`account_name`

Cloud Platform/region	Full Account Name
Canada (Central)	account_name.ca-central-1.aws
EU (Ireland)	account_name.eu-west-1
EU (Frankfurt)	account_name.eu-central-1
Asia Pacific (Singapore)	account_name.ap-southeast-1
Asia Pacific (Sydney)	account_name.ap-southeast-2
Asia Pacific (Tokyo)	account_name.ap-northeast-1.aws
Azure	
East US 2	account_name.east-us-2.azure
West US 2	account_name.west-us-2.azure
US Gov Virginia	account_name.us-gov-virginia.azure
Canada Central	account_name.canada-central.azure
West Europe	account_name.west-europe.azure
Australia East	account_name.australia-east.azure
Southeast Asia	account_name.southeast-asia.azure

Default Value

No default value

Arrow Fallback To JSON

Attribute

ArrowFallbackToJson (aftj)

Purpose

Specifies whether the driver uses the JSON query format when the arrow format is not properly initialized. The driver uses a high-speed arrow transfer that requires the restricted APIs from java.nio package. For JVMs that are Java SE 16 (or equivalent) and higher, the following JVM value must be set to use arrow transfer:

`--add-opens=java.base/java.nio=ALL-UNNAMED`

If this value is not specified, the API throws an `InaccessibleObjectException` during query execution. To allow you to continue executing queries in this scenario, you can set this option to configure the driver to fall back to the JSON query format when the JVM is not properly configured.

Valid Values

`0` | `1` | `2`

Behavior

If set to `0` (Enable), the driver falls back to JSON query format if the arrow APIs can not be initialized using Java SE 16 and higher, and a CONFIG-level message is logged.

If set to `1` (Warn), the driver falls back to JSON query format if the arrow APIs can not be initialized using Java SE 16 and higher, and a WARN-level message is logged.

If set to `2` (Disable), the driver does not fall back to JSON query format if arrow APIs can not be initialized, and the driver throws a `SQLException` when executing a `SELECT` query that uses arrow transfer.

Notes

- The recommended practice is to configure the JVM with the `--add-opens=java.base/java.nio=ALL-UNNAMED` option value and disable this connection option. This configuration provides the best performance.
- Alternatively, your application can set JSON query format for the session by executing the `ALTER SESSION SET JDBC_QUERY_RESULT_FORMAT='JSON'` statement.
- This connection option can affect performance.

Default Value

`0` (Enable)

See also

Performance considerations on page 62

Authorization Code

Attribute

OAuthCode (oac)

Purpose

Specifies the temporary authorization code that is exchanged for access tokens when OAuth 2.0 authentication is enabled.

Valid Values

String

where:

String

is an OAuth 2.0 authorization code.

Notes

- This option is configured by the application.
- See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Authentication Method

Attribute

AuthenticationMethod (am)

Purpose

Specifies the authentication method to use for verifying user login credentials.

Valid Values

`0` | `24` | `21` | `45`

Behavior

If set to `0` (UserIDPassword), the driver uses a user ID and password when establishing a connection.

If set to `24` (OAuth2), the driver uses OAuth2 authentication, a token-based authentication when establishing a connection.

If set to `45` (BrowserBasedSSO), the driver uses a web browser, ADFS, or any other SAML 2.0-compliant identify provider (IdP) when establishing a connection.

If set to `21` (KeyPair), the driver uses a pair of private and public keys for authentication.

Default Value

`0` (UserIDPassword)

See also

Authentication on page 43

Authorization URI

Attribute

AuthURI (o2au)

Purpose

Specifies the endpoint for obtaining an authorization code from a third-party authorization service for OAuth 2.0 implementations.

Valid Values

String

where:

String

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

Notes

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

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Client ID

Attribute

ClientID (cid)

Purpose

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

Valid Values

String

where:

String

is the client ID key for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Client Secret

Attribute

ClientSecret (clse)

Purpose

Specifies the client secret for your application when authenticating to Snowflake with OAuth 2.0 enabled.

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

Valid Values

`String`

where:

`String`

is the client secret for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Client Session Keep Alive

Attribute

ClientSessionKeepAlive (cska)

Purpose

Specifies whether to keep the current session active after a period of inactivity or to force the user to login again.

Valid Values

`0` | `1`

Behavior

If set to `1` (true), the session remains active indefinitely, even if there is no activity from the user.

If set to `0` (false), the session is closed after four hours of inactivity. The user must re-authenticate to start a new session.

Notes

- Closing the session during periods of inactivity causes the warehouse to consume less credits.

Default Value

`0` (false)

Data Source Name

Attribute

DataSourceName (dsn)

Purpose

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

Valid Values

`String`

where:

`String`

is the name of a data source.

Default Value

No default value

Database Name

Attribute

DatabaseName (db)

Purpose

Specifies the name of the database to which you want to connect.

Valid Values

database_name

where:

database_name

is the name of a valid database.

Notes

- The default role should have privileges for connecting to the specified database.

Default Value

No default value

Description

Attribute

Description (desc)

Purpose

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

Valid Values

String

where:

String

is a description of a data source.

Default Value

No default value

Disable SOCKS Proxy

Attribute

DisableSocksProxy (dsp)

Purpose

Specifies whether the driver should ignore the SOCKS proxy configuration specified in the Java system options (if any). Setting this connection property alters the behavior of all the connections on the same JVM.

Valid Values

`0` | `1`

Behavior

If set to `0` (false), the driver uses the SOCKS proxy.

If set to `1` (true), the driver ignores the SOCKS proxy.

Notes

This property is used when you want the driver to ignore the existing Java proxy configuration on your client for Snowflake connections.

Default Value

`0` (false)

Extended Options

Attribute

ExtendedOptions (xo)

Purpose

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

Valid Values

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

where:

option

is a connection option.

value

is the value or setting of the connection option.

Default Value

No default value

Integer Field Mapping

Attribute

IntegerFieldMapping (ifm)

Purpose

Specifies the data type mapping for fixed-point numbers where precision and scale cannot be specified. These include BIGINT, INTEGER, and synonymous types. Refer to your Snowflake documentation for details.

Valid Values

`0` | `1`

Behavior

If set to `0` (BigInt), the driver maps fixed-point numbers where precision and scale cannot be specified to BigInt.

If set to `1` (Numeric), the driver maps fixed-point numbers where precision and scale cannot be specified to Numeric.

Default Value

`1` (Numeric)

JVM Arguments

Attribute

JVMArgs (jvma)

Purpose

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

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

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

Valid Values

`String`

where:

String

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

Example

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

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

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

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

Default Value

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

```
-Xmx256m
```

For all other configurations:

```
-Xmx1024m
```

See also

Performance considerations on page 62

JVM Classpath

Attribute

JVMClassPath (jvmcp)

Purpose

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

Valid Values

string

where:

string

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

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

Example

On Windows:

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

On Linux:

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

Default Value

No default value

JVM Path

Attribute

JVMPath (jvmp)

Purpose

Specifies fully qualified path to the JVM executable that you want to use to run the SQL Engine Server. The path must not contain double quotation marks.

Valid Values

String

where:

String

The full path to the JVM executable.

Default Value

install_dir`\jre\bin\java.exe`

Log Config File

Attribute

LogConfigFile (lgcf)

Purpose

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

Valid Values

String

where:

String

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

Default Value

`ddlogging.properties`

Login Timeout

Attribute

LoginTimeout (lt)

Purpose

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

Valid Values

`-1` | `0` | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to `-1`, the connection request does not time out. The driver silently ignores the `SQL_ATTR_LOGIN_TIMEOUT` attribute.

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

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

Default Value

`60`

Partner Application Name

Attribute

PartnerApplicationName (pan)

Purpose

For Snowflake partner use only.

Specifies the name of a partner application to which you are trying to connect. This option is useful for users who have an existing partner contract with Snowflake and are using the driver with a Snowflake partner application or plug-in.

Valid Values

String

where:

String

is a valid partner application name.

Default Value

No default value

Password

Attribute

Password (pwd)

Purpose

A password that is used to connect to the instance.

Behavior

password

where:

password

is a valid password. The password is case-sensitive.

Default Value

No default value

See also

Authentication on page 43

Private Key Content

Attribute

PrivateKeyContent (pkc)

Purpose

Specifies the content of a private key. It is used to authenticate to Snowflake when key-pair authentication is enabled (`AuthenticationMethod=KeyPair`).

Valid Values

`String`

where:

`String`

is the content of a private key.

Notes

- When specifying private key content, use the following syntax:
  - For encrypted private key content: `-----BEGIN ENCRYPTED PRIVATE KEY-----`*`private key content`*`-----END ENCRYPTED PRIVATE KEY-----`
  - For unencrypted private key content: `-----BEGIN PRIVATE KEY-----`*`private key content`*`-----END PRIVATE KEY-----`
- If the encryption schema you are using to generate the encrypted private keys is not compatible with the native encryption libraries of your JRE, the JRE will return an error. To resolve this issue, either generate encrypted private keys using an encryption schema that is compatible with your JRE or add a third-party Java cryptography library to your application (for example, Bouncy Castle) that supports the encryption schema you are using. For example, to add Bouncy Castle to your application, add the following line to the `java.security` file: `security.provider.`*`n`*`=org.bouncycastle.jce.provider.BouncyCastleProvider`, and then add the Bouncy Castle jars to the classpath of your JRE.
- The private key is a confidential value. To prevent unauthorized access, this value must be securely maintained.
- The private key content you specify must correspond to the content of the public key assigned to your Snowflake account.

Default Value

No default value

See also

Authentication on page 43

Private Key File

Attribute

PrivateKeyFile (pkf)

Purpose

Specifies the absolute path to the private key file used to authenticate to Snowflake when key-pair authentication is enabled (`AuthenticationMethod=KeyPair`).

Valid Values

`String`

where:

`String`

is the the absolute path to the private key file.

Notes

- If the encryption schema you are using to generate the encrypted private keys is not compatible with the native encryption libraries of your JRE, the JRE will return an error. To resolve this issue, either generate encrypted private keys using an encryption schema that is compatible with your JRE or add a third-party Java cryptography library to your application (for example, Bouncy Castle) that supports the encryption schema you are using. For example, to add Bouncy Castle to your application, add the following line to the `java.security` file: `security.provider.`*`n`*`=org.bouncycastle.jce.provider.BouncyCastleProvider`, and then add the Bouncy Castle jars to the classpath of your JRE.
- The private key is a confidential value. To prevent unauthorized access, this value must be securely maintained.
- The private key contained in the private key file you specify must correspond to the public key assigned to your Snowflake account.

Default Value

No default value

See also

Authentication on page 43

Private Key Passphrase

Attribute

PrivateKeyPassphrase (pkpp)

Purpose

Specifies the password used to decrypt the encrypted private key. The private key is used to authenticate to Snowflake when key-pair authentication is enabled (`AuthenticationMethod=KeyPair`).

Valid Values

`String`

where:

`String`

is the password used to decrypt the encrypted private key.

Notes

- This option can be used with both the PrivateKeyFile and PrivateKeyContent connection options, to decrypt either a private key or the content of a private key.
- Both private key and private key password are confidential values. To prevent unauthorized access, these values must be securely maintained.

Default Value

No default value

See also

Authentication on page 43

Proxy Host

Attribute

ProxyHost (pxhn)

Purpose

Identifies a proxy server to use for the first connection.

Valid Values

`server_name` | *`IP_address`*

where:

`server_name`

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

`IP_address`

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

Default Value

No default value

See also

Proxy server support on page 60

Proxy Password

Attribute

ProxyPassword (pxpw)

Purpose

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

Valid Values

password

where:

password

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

Default Value

No default value

See also

Proxy server support on page 60

Proxy Port

Attribute

ProxyPort (pxpt)

Purpose

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

Valid Values

port

where:

port

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

Default Value

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

For HTTP: `80`

For HTTPS: `443`

See also

Proxy server support on page 60

Proxy User

Attribute

ProxyUser (pxun)

Purpose

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

Valid Values

`user_name`

where:

`user_name`

is a valid user ID for the proxy server.

Default Value

No default value

See also

Proxy server support on page 60

Query Timeout

Attribute

QueryTimeout (qt)

Purpose

Sets the default query timeout (in seconds) for all statements created by a connection.

Valid Values

`-1` | `0` | *`x`*

where:

x

is a number of seconds.

Behavior

If set to `-1`, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute.

If set to `0`, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute.

If set to *x*, all queries time out after the specified number of seconds unless the application overrides this value by setting the SQL_ATTR_QUERY_TIMEOUT attribute.

Default Value

`0`

Redirect URI

Attribute

RedirectURI (o2ru)

Purpose

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

Valid Values

`String`

where:

`String`

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

Notes

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

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Refresh Token

Attribute

RefreshToken (rtok)

Purpose

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

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

Valid Values

String

where:

String

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

Notes

- See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (rcce)

Purpose

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

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

Valid Values

`0 | 1 | 2`

Behavior

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

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

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

Default Value

`0`

Role Name

Attribute

RoleName (role)

Purpose

Specifies the default role to use for access control in the Snowflake session initiated by the driver. The specified role should be an existing role that has already been assigned to the specified user. If the specified role has not already been assigned to the user, the role is not used when the session is initiated by the driver.

Valid Values

`string`

where:

`string`

is name of the default role to use for access control.

Notes

- The USE ROLE command can be executed to set a different role for the session.

Default Value

No default value

Schema

Attribute

Schema (schm)

Purpose

Specifies the default schema to use for the specified database once connected. The specified schema should be an existing schema for which the specified default role has privileges.

After connecting, the USE SCHEMA command can be executed to set a different schema for the session.

Valid Values

String

where:

String

is a valid schema name.

Default Value

No default value

Scope

Attribute

Scope (oas)

Purpose

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

Valid Values

String

where:

String

is a space-separated list of security scopes.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Server Port Number

Attribute

ServerPortNumber (sport)

Purpose

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

Valid Values

port_number

where:

port_number

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

Notes

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

Default Value

32-bit driver: 19948

64-bit driver: 19947

See also

SQL Engine Mode on page 102

SQL Engine Mode

Attribute

SQLEngineMode (sem)

Purpose

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

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

Default Value

For Windows:

1 (Server)

For Linux:

2 (Direct)

See also

Using the SQL engine server on page 63

SQL Service

Attribute

SQLService (ss)

Purpose

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

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

Default Value

No default value

Token URI

Attribute

TokenURI (o2tu)

Purpose

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

Valid Values

String

where:

String

is the endpoint used to retrieve access tokens.

Notes

See "OAuth 2.0 authentication" for more information.

Default Value

No default value

See also

OAuth 2.0 authentication on page 48

Use Session Database for Metadata

Attribute

UseSessionDatabaseForMetadata (usdm)

Purpose

Specifies whether the driver fetches metadata for only tables in the database to which you are connected when a database name is not specified in metadata calls. When enabled, this option can provide better performance for metadata calls by reducing the number of tables queried.

Valid Values

`0` | `1`

Behavior

If set to `1` (true), the driver fetches metadata for only tables in the database to which you are connected.

If set to `0` (false), the driver fetches metadata for all tables in all databases.

Notes

- When the database name is specified in the metadata calls, the driver ignores the value specified for this connection option and returns metadata only for the database specified in the metadata calls.

Default Value

`0` (false)

User

Attribute

User

Purpose

Specifies the user name that is used to connect to the instance.

Valid Values

`String`

where:

String

is a valid user name. The user name is case-insensitive.

Default Value

No default value

See also

Authentication on page 43

Warehouse

Attribute

Warehouse (wh)

Purpose

Specifies the virtual warehouse to use once connected. The specified warehouse should be an existing warehouse for which the specified default role has privileges.

After connecting, the USE WAREHOUSE command can be executed to set a different warehouse for the session.

Valid Values

String

where:

String

is a valid warehouse string.

Default Value

No default value