Progress DataDirect

Progress DataDirect for ODBC for Amazon Redshift Wire Protocol Driver

User's Guide

Release 8.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Amazon Redshift Wire Protocol Driver

The Progress® DataDirect® for ODBC™ for Amazon Redshift™ Wire Protocol driver supports standard SQL query language to access data managed by Amazon Redshift. The driver is supported in the Windows, UNIX, and Linux environments.

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

For details, see the following topics:

- Contacting Technical Support

What's new in this release?

Support and certification

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

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

Changes since 8.0.0

- **Driver Enhancements**
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The Show Selectable Tables (ShowSelectableTables) connection option has been added to the driver. It determines whether the driver returns metadata for all tables or only those for which the user has Select privileges. It can help the users with restricted select privileges retrieve metadata for the required tables. See Show Selectable Tables on page 135 for details.
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: `xxopenssl30.dll` (for Windows) and `xxopenssl30.so[.sl]` (for UNIX/Linux).

    The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL server authentication on page 76 and TLS/SSL client authentication on page 79 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 63 for details.
  - The driver has been enhanced to support Azure Active Directory (Azure AD) authentication. It allows administrators to centrally manage user permissions to Amazon Redshift. When Azure AD authentication is enabled, all communications to Amazon Redshift are encrypted. See Azure Active Directory authentication on page 86 for details.
  - The driver has been enhanced to support the following data types: Float, Tinyint, Wchar, and Wvarchar.
  - The new AllowedOpenSSLVersions option allows you to determine which version of the OpenSSL library file the driver uses for data encryption.

- **Changed Behavior**
  - The TLSv1.1 and TLSv1.0 cryptographic protocols are now disabled by default and have been removed as selectable values for the Crypto Protocol Version (CryptoProtocolVersion) option on the Setup dialog. These protocols are no longer considered secure and, therefore, are no longer recommended for use. However, the driver still supports TLSv1.1 and TLSv1.0 for legacy servers that do not support more secure protocols. See Crypto Protocol Version on page 108 for more information.

- The Allowed OpenSSL Versions (AllowedOpenSSLVersions) connection option has been deprecated as the driver currently supports only version 3.0 of the OpenSSL library.
- The product no longer includes version 1.1.1 of the OpenSSL library. The library will reach the end of its product life cycle in September 2023 and will not receive any security updates after that. Note that continuing to use the library after September 2023 can potentially expose you to security vulnerabilities.

  Note: As a result of this change, when installing a new version of the product, the installer program will automatically remove version 1.1.1 of the library from the install directory, which will impact all the DataDirect ODBC drivers installed on a machine. Therefore, if you are using multiple drivers, upgrade all your drivers to the latest version.

- The product no longer includes version 1.0.2 of the OpenSSL library. The library has reached the end of its product life cycle and is not receiving security updates anymore. Note that continuing to use the library could potentially expose you to security vulnerabilities.

  Note: As a result of this change, when installing a new version of the driver, the installer program will automatically remove version 1.0.2 of the library from the install directory.

- The crypto protocol versions prior to TLSv1 are no longer supported.

Changes for 8.0.0 GA

- **Driver Enhancements**
  - The driver is now compiled using Visual Studio 2015 for improved security.
  - Support for connecting to a proxy server through an HTTP connection. HTTP proxy support is configurable with five new connection options. See Proxy Host, Proxy Mode, Proxy Password, Proxy Port, and Proxy User for details.
  - The driver has been enhanced to support the TimestampTZ data type. See Data types on page 15 and Fetch TSWTZ as Timestamp on page 117for details.
  - The new Fetch TSWTZ as Timestamp option allows you to determine whether the driver returns column values of the TimestampTZ data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR. See Fetch TSWTZ as Timestamp on page 117 for details.
  - The driver has been enhanced to support the HOUR, MINUTE, MONTH, QUARTER, SECOND, WEEK, and YEAR ODBC functions for improved support of third-party applications such as Tableau.
  - The driver includes a new Tableau data source file (Windows only) that provides improved functionality when accessing your data with Tableau. See Accessing data in Tableau (Windows only) on page 27 for details.
  - The driver and Driver Manager have been enhanced to support UTF-8 encoding in the `odbc.ini` and `odbcinst.ini` files.

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

- **Changed Behavior**
  - The default value for Crypto Protocol Version has been updated to `TLSv1.2,TLSv1.1,TLSv1`. This change improves the security of the driver by employing only the most secure cryptographic protocols as the default behavior. See Crypto Protocol Version on page 108 for details.

Driver requirements

Data source and platform requirements

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

AIX requirements for 32-bit and 64-bit drivers

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

Oracle Solaris requirements for 32-bit drivers

- The following processors are supported:
  - Oracle SPARC
  - x86: Intel
  - x64: Intel and AMD
- For Oracle SPARC: An application compatible with components that were built using Oracle Workshop version 6 update 2 and the Solaris native (kernel) threading model.
- For x86/x64: An application compatible with components that were built using Oracle C++ 5.8 and the Solaris native (kernel) threading model.

Oracle Solaris requirements for 64-bit drivers

- The following processors are supported:
  - Oracle SPARC
  - x64: Intel and AMD
- For Oracle SPARC: An application compatible with components that were built using Oracle Workshop version 6 update 2 and the Solaris native (kernel) threading model.
- For x64: An application compatible with components that were built using Oracle C++ Compiler version 5.8 and the Solaris native (kernel) threading model.

ODBC compliance

The driver is compliant with the Open Database Connectivity (ODBC) specification. The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions.

In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLTablePrivileges

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for additional information.

Version string information

The driver has a version string of the format:

```
XX.YY.ZZZZ(BAAAA, UBBBB)
```

or

```
XX.YY.ZZZZbAAAA, uBBBB)
```

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

```
XX.YY.ZZZZ(UBBBB)
```

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.

For example:

```
08.00.0002 (B0001, U0002)
      |__|  |___|  |___|
     Driver  Bas    Utl
```

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 Version tab, click **File Version** in the Other version information list box.

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 drives 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 ivrsft28.so
```

returns:

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

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 32-bit ICU component on Linux:

```
ivtestlib libivicu28.so
08.00.0001
```

Note: Only the HP-UX version of the tool requires specifying the full path for the test loading tool. The full path does not need to be specified for other platforms.

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 shows how the Amazon Redshift data types are mapped to the standard ODBC data types.

Table 1: Amazon Redshift Data Types

Amazon Redshift	ODBC
Bigint	SQL_BIGINT
Boolean	SQL_BIT
Character	SQL_CHAR
Varchar	SQL_VARCHAR
Date	SQL_TYPE_DATE
Decimal[1]	SQL_NUMERIC
Double Precision	SQL_DOUBLE
Float	SQL_REAL

[1] Numeric is an alias for Decimal.

Amazon Redshift	ODBC
Integer	SQL_INTEGER
Real	SQL_REAL
Smallint	SQL_SMALLINT
Timestamp	SQL_TYPE_TIMESTAMP
TimestampTZ	SQL_VARCHAR | SQL_TYPE_TIMESTAMP[2]
Tinyint	SQL_SMALLINT
Wchar	SQL_CHAR
Wvarchar	SQL_VARCHAR

Retrieving data type information

At times, you might need to get information about the data types that are supported by the data source, for example, precision and scale. You can use the ODBC function SQLGetTypeInfo to do this.

On Windows, you can use ODBC Test to call SQLGetTypeInfo against the ODBC data source to return the data type information.

Refer to "Diagnostic tools" in the *Progress DataDirect for ODBC Drivers Reference* for details about ODBC Test.

[2] TimestampTZ mapping changes based on the setting of the Fetch TSWTZ as Timestamp option. The default is SQL_VARCHAR.

On all platforms, an application can call SQLGetTypeInfo. Here is an example of a C function that calls SQLGetTypeInfo and retrieves the information in the form of a SQL result set.

```
void ODBC_GetTypeInfo(SQLHANDLE hstmt, SQLSMALLINT dataType)
{
    RETCODE rc;

// There are 19 columns returned by SQLGetTypeInfo.
// This example displays the first 3.
// Check the ODBC 3.x specification for more information.
// Variables to hold the data from each column
    char             typeName[30];
    short            sqlDataType;
    unsigned int     columnSize;

    SQLLEN           strlenTypeName,
                     strlenSqlDataType,
                     strlenColumnSize;

    rc = SQLGetTypeInfo(hstmt, dataType);
    if (rc == SQL_SUCCESS) {

// Bind the columns returned by the SQLGetTypeInfo result set.
        rc = SQLBindCol(hstmt, 1, SQL_C_CHAR, &typeName,
              (SDWORD)sizeof(typeName), &strlenTypeName);
        rc = SQLBindCol(hstmt, 2, SQL_C_SHORT, &sqlDataType,
              (SDWORD)sizeof(sqlDataType), &strlenSqlDataType);
        rc = SQLBindCol(hstmt, 3, SQL_C_LONG, &columnSize,
              (SDWORD)sizeof(columnSize), &strlenColumnSize);

// Print column headings
         printf ("TypeName      DataType          ColumnSize\n");
         printf ("-------------------- ---------- ----------\n");

        do {

// Fetch the results from executing SQLGetTypeInfo
            rc = SQLFetch(hstmt);
            if (rc == SQL_ERROR) {
// Procedure to retrieve errors from the SQLGetTypeInfo function
                ODBC_GetDiagRec(SQL_HANDLE_STMT, hstmt);
                break;
               }

// Print the results
         if ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO)) {
printf ("%-30s %10i %10u\n", typeName, sqlDataType, columnSize);
             }

        } while (rc != SQL_NO_DATA);
    }
}
```

SQL support

The driver supports the core SQL grammar.

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.

October 2017, Release 8.0.0 of the Progress DataDirect for ODBC for Amazon Redshift Wire Protocol Driver, Version 0001

2

Getting started

This chapter provides basic information about configuring your driver immediately after installation and testing your connection. To take full advantage of the features of the driver, read "Using the Driver."

Information that the driver needs to connect to a database is stored in a *data source*. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on UNIX/Linux), and file data sources. On Windows, user and system data sources are stored in the registry of the local computer. The difference is that only a specific user can access user data sources, whereas any user of the machine can access system data sources. On Windows, UNIX, 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.

When you define and configure a data source, you store default connection values for the driver that are used each time you connect to a particular database. You can change these defaults by modifying the data source.

For details, see the following topics:

Configuring and connecting on Windows

The following basic information enables you to configure a data source and test connect with a driver immediately after installation. On Windows, you can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box. Default connection values are specified through the options on the tabs of the Setup dialog box and are stored either as a user or system data source in the Windows Registry, or as a file data source in a specified location.

Configuring a data source

To configure a data source:

1. From the Progress DataDirect program group, start the ODBC Administrator and click either the **User DSN**, **System DSN**, or **File DSN** tab to display a list of data sources.

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

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

   - **System DSN**: To configure a new system data source, click **Add** to display a list of installed drivers. Select the appropriate driver and click **Finish** to display the driver Setup dialog box.
   - **File DSN**: To configure a new file data source, click **Add** to display a list of installed drivers. Select the driver and click **Advanced** to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

   The General tab of the Setup dialog box appears by default.

   Note: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise in this book.

2. On the General tab, provide the following information; then, click **Apply**.

   Host Name: Type the IP address of the interface to which you want to connect.

   Port Number: Type the port number of the server listener. The default is `5439`.

   Database Name: Type the name of the database to which you want to connect.

Testing the connection

To test the connection:

1. After you have configured the data source, you can click **Test Connect** on the Setup dialog box to attempt to connect to the data source using the connection options specified in the dialog box. The driver returns a message indicating success or failure. A logon dialog box appears as described in "Using a logon dialog box."
2. Supply the requested information in the logon dialog box and click **OK**. Note that the information you enter in the logon dialog box during a test connect is not saved.

   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.

- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

3. On the driver Setup dialog box, click **OK**. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the previously described procedure to modify your data source. You can override these defaults by connecting to the data source using a connection string with alternate values. See "Using a logon dialog box" for information about using connection strings.

See also

Using a logon dialog box on page 64

Configuring and connecting on UNIX and Linux

UNIX®

The following basic information enables you to configure a data source and test connect with a driver immediately after installation. See "Configuring and connecting to data sources" for detailed information about configuring the UNIX/Linux environment and data sources.

Note: In the following examples, *xx* in a driver filename represents the driver level number.

See also

Configuring and connecting to data sources on page 35

Environment configuration

To configure the environment:

1. Check your permissions: You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.
2. From your login shell, determine which shell you are running by executing:

   ```
   echo $SHELL
   ```

3. 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`. After running the setup script, execute:

   ```
   env
   ```

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

4. Set the ODBCINI environment variable. The variable must point to the path from the root directory to the system information file where your data source resides. The system information file can have any name, but the product is installed with a default file called `odbc.ini` in the product installation directory. For example, if you use an installation directory of `/opt/odbc` and the default system information file, from the Korn or Bourne shell, you would enter:

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

From the C shell, you would enter:

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

Test loading the driver

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

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

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

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

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

Configuring a data source in the system information file

The default `odbc.ini` file installed in the installation directory is a template in which you create data source definitions. You enter your site-specific database connection information using a text editor. Each data source definition must include the keyword `Driver=`, which is the full path to the driver.

The following examples show the minimum connection string options that must be set to complete a test connection, where *xx* represents iv for 32-bit or dd for 64-bit drivers, *yy* represents the driver level number, and *zz* represents the extension. The values for the options are samples and are not necessarily the ones you would use.

```
[ODBC Data Sources]
Amazon Redshift Wire Protocol=DataDirect 8.0 Amazon Redshift Wire Protocol

[Amazon Redshift Wire Protocol]
Driver=ODBCHOME/lib/xxrsftyy.zz
Database=Redshiftdb1
HostName=RedshiftServer
PortNumber=5439
```

Connection option descriptions:

`Database`: The name of the database to which you want to connect by default.

`HostName`: Either the name or the IP address of the server to which you want to connect.

`Port Number`: The port number of the server listener. The default is `5439`.

Testing the connection

The driver installation includes an ODBC application called example that can be used to connect to a data source and execute SQL. The application is located in the `*installation_directory*/samples/example` directory.

To run the program after setting up a data source in the `odbc.ini`, enter `example` and follow the prompts to enter your data source name, user name, and password. If successful, a `SQL>` prompt appears and you can type in SQL statements such as `SELECT * FROM *table*`. If example is unable to connect, the appropriate error message is returned.

3

Tutorials

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

- Accessing data in Tableau (Windows only) on page 27
- Accessing data in Microsoft Excel from the Data Connection Wizard (Windows only) on page 30
- Accessing data in Microsoft Excel from the Query Wizard (Windows only) on page 33

For details, see the following topics:

- Accessing data in Tableau (Windows only)
- Accessing data in Microsoft Excel from the Data Connection Wizard (Windows only)
- Accessing data in Microsoft Excel from the Query Wizard (Windows only)

Accessing data in Tableau (Windows only)

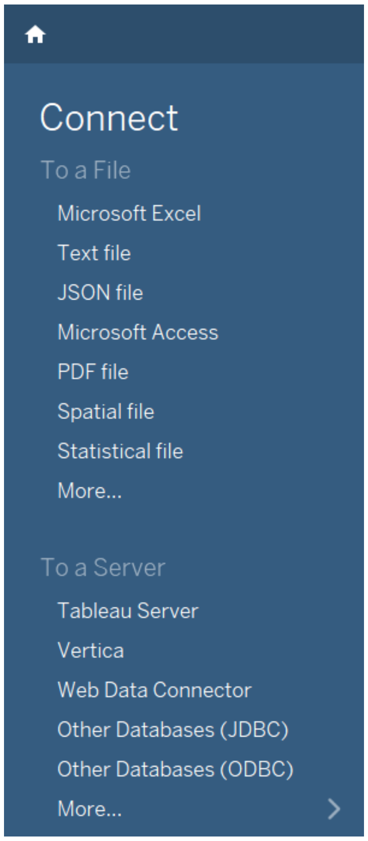
After you have configured your data source, you can use the driver to access your Amazon Redshift data with Tableau. Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Tableau, you can improve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

To use the driver to access data with Tableau:

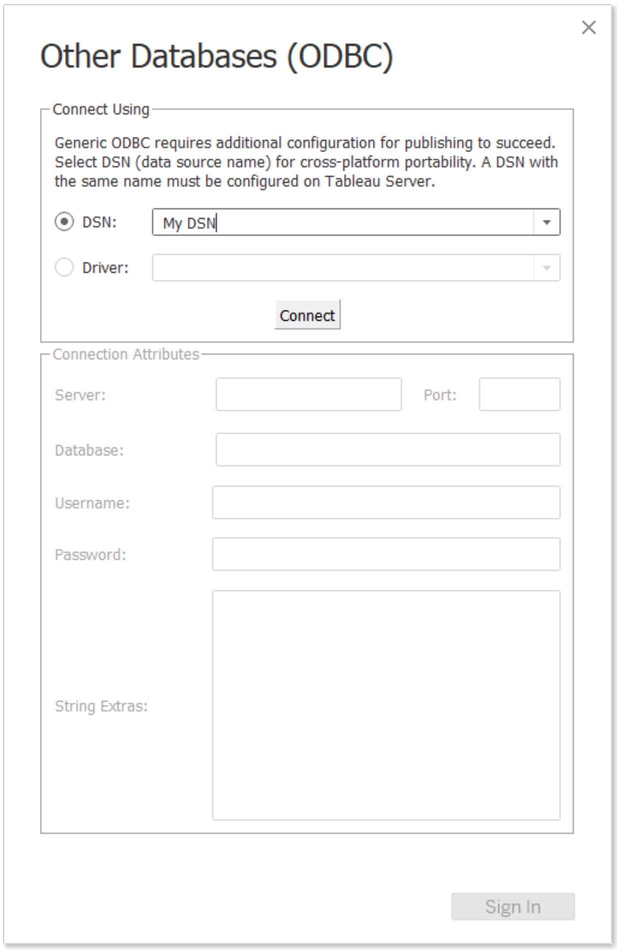
1. Navigate to the `\tools\Tableau` subdirectory of the Progress DataDirect installation directory; then, locate the Tableau data source file, `DataDirect Redshift.tdc`.
2. Copy the `DataDirect Redshift.tdc` file into the following directory:

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

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



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



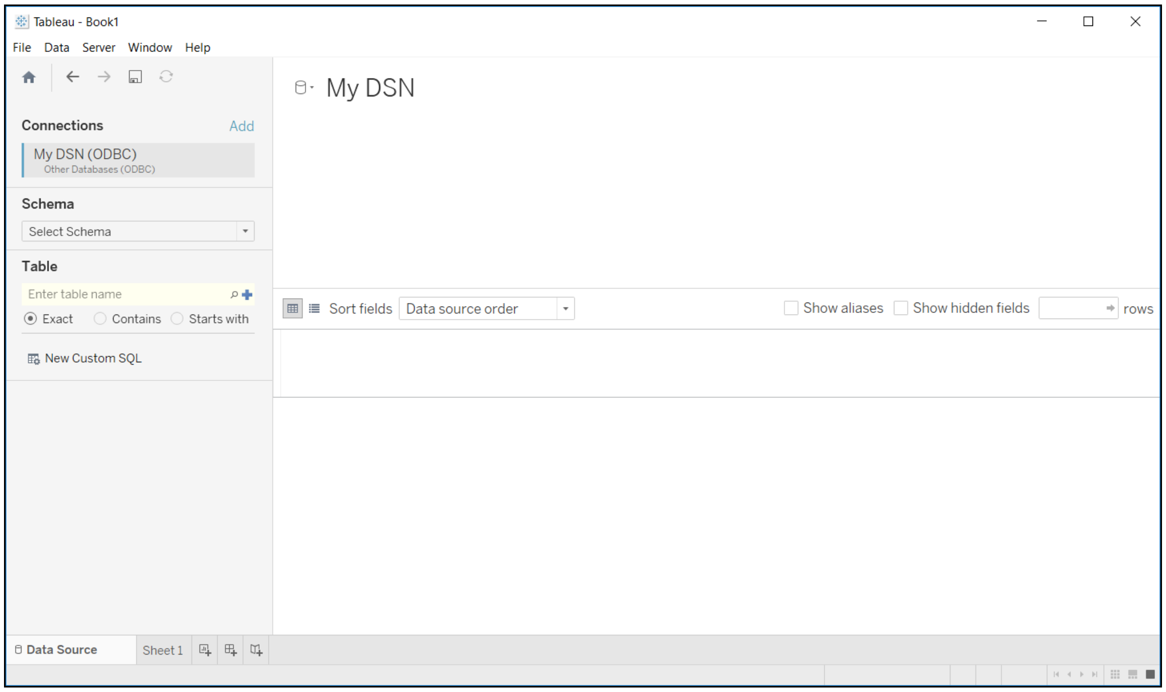
Generic ODBC requires additional configuration for publishing to succeed. Select DSN (data source name) for cross-platform portability. A DSN with the same name must be configured on Tableau Server.

[✓ DSN:] My DSN

Driver:

In the DSN field, select the data source you want to use from the drop down menu. For example, **My DSN**. Then, click **Connect**. The **Logon to Amazon Redshift** dialog appears pre-populated with the connection information you provided in your data source.

6. If required, type your user name and password; then, click **OK**. The Logon dialog closes. Then, click **OK** on the Server Connection dialog.
7. The **Data Source** window appears.



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

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

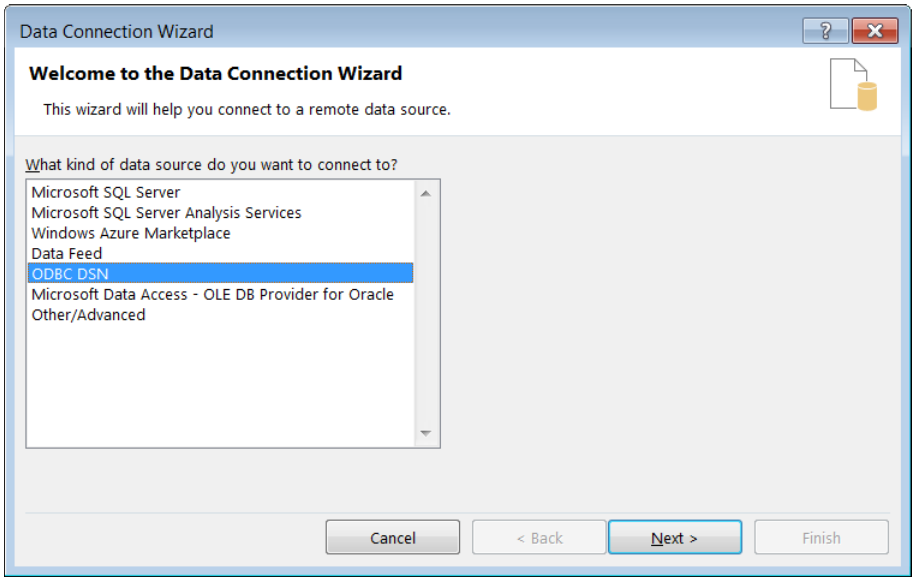
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

Accessing data in Microsoft Excel from the Data Connection Wizard (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 **From Other Sources**>**From Data Connection Wizard**.
3. The **Welcome to Data Connection Wizard** dialog appears.

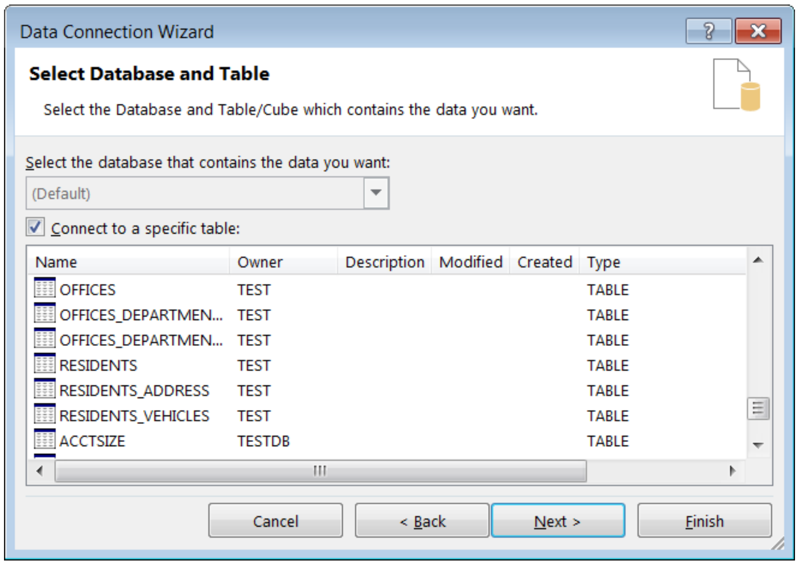

Data Connection Wizard
Welcome to the Data Connection Wizard
This wizard will help you connect to a remote data source.
What kind of data source do you want to connect to?
Microsoft SQL Server
Microsoft SQL Server Analysis Services
Windows Azure Marketplace
Data Feed
ODBC DSN
Microsoft Data Access - OLE DB Provider for Oracle
Other/Advanced
Cancel
< Back
Next >
Finish

Select **ODBC DSN** from the data source list; then, click **Next**.

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

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

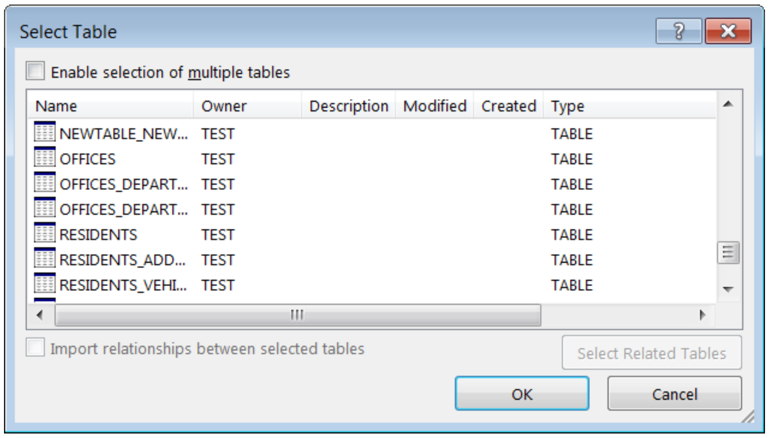
6. The **Select Database and Table** window appears.


Data Connection Wizard
Select Database and Table
Select the Database and Table/Cube which contains the data you want.
Select the database that contains the data you want:
(Default)
Connect to a specific table:
Cancel
< Back
Next >
Finish

Name	Owner	Description	Modified	Created	Type
OFFICES	TEST				TABLE
OFFICES_DEPARTMEN...	TEST				TABLE
OFFICES_DEPARTMEN...	TEST				TABLE
RESIDENTS	TEST				TABLE
RESIDENTS_ADDRESS	TEST				TABLE
RESIDENTS_VEHICLES	TEST				TABLE
ACCTSIZE	TESTDB				TABLE

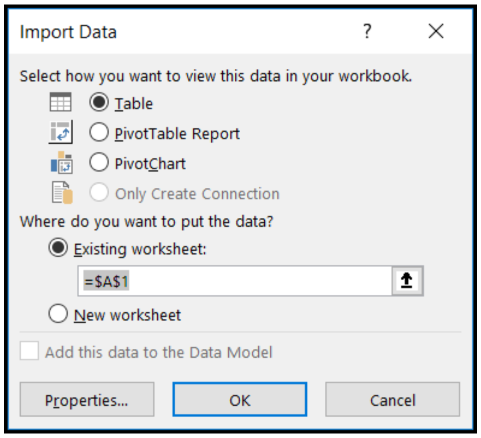
- To access data from multiple tables, uncheck the **Connect to a specific table** box; then, click **Next**.
- To access data from a specific table, select the table you want to import from the list; then, click **Next**.

7. Enter the file name, description, and, optionally, the friendly name and search keywords for your Data Connection file in the corresponding fields. Click **Finish**.
   - If you chose to access data from multiple tables in Step 6 on page 31, proceed to the next step.
   - If you chose to access data from a specific table in Step 6 on page 31, skip to step 9 on page 32.
8. The Select Table window appears.


Select Table
Enable selection of multiple tables
Import relationships between selected tables
Select Related Tables
OK
Cancel

Name	Owner	Description	Modified	Created	Type
NEWTABLE_NEW...	TEST				TABLE
OFFICES	TEST				TABLE
OFFICES_DEPART...	TEST				TABLE
OFFICES_DEPART...	TEST				TABLE
RESIDENTS	TEST				TABLE
RESIDENTS_ADD...	TEST				TABLE
RESIDENTS_VEHI...	TEST				TABLE

   Select the tables that you want to import into your workbook; then, click **OK**.
9. The Import Data window appears.



Select how you want to view this data in your workbook.
[✓ Table]
PivotTable Report
PivotChart
Only Create Connection
Where do you want to put the data?
[✓ Existing worksheet:]
=A1
New worksheet
Add this data to the Data Model

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

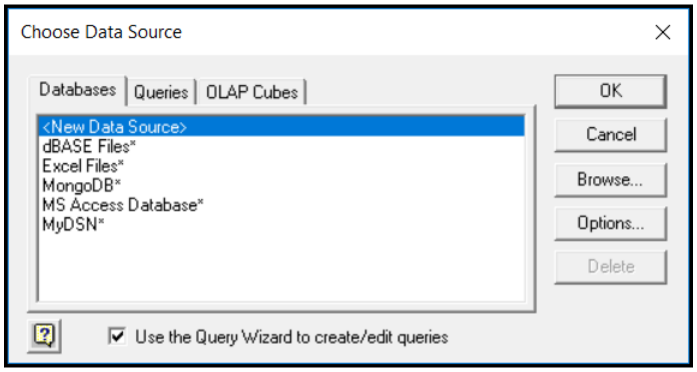
You have successfully accessed your data in Excel using the Data Connection Wizard. For more information, refer to the Microsoft Excel product documentation at: https://support.office.com/.

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

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

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

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



<New Data Source>
dBASE Files*
Excel Files*
MongoDB*
MS Access Database*
MyDSN*

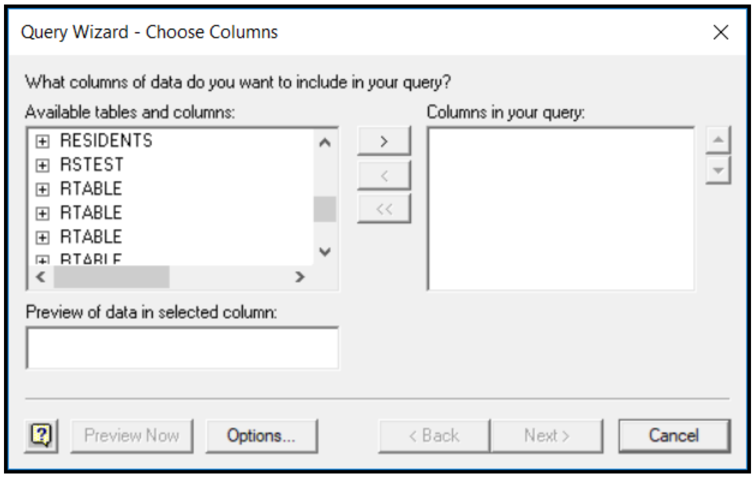
☑ Use the Query Wizard to create/edit queries

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

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

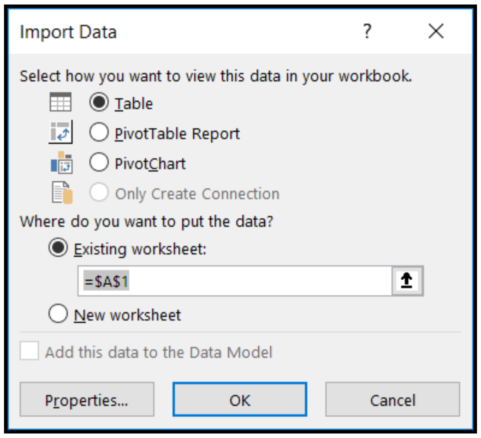
Note: The logon dialog may reappear if Excel needs to access additional information from the data source. If this occurs, re-enter your password; then, click **OK** to proceed to the next step.

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


Query Wizard - Choose Columns
What columns of data do you want to include in your query?
Available tables and columns:
RESIDENTS
RSTEST
RTABLE
RTABLE
RTABLE
RTABLE
Columns in your query:
Preview of data in selected column:
Preview Now
Options...
< Back
Next >
Cancel

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

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



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

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

4

Using the driver

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

For details, see the following topics:

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. See "Getting started" for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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

See also

Getting started on page 21

Using a connection string on page 62

Configuring the product on UNIX/Linux

UNIX®

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

See "Environment variables" for additional platform information.

See also

Environment variables on page 36

Environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

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

- `LD_LIBRARY_PATH` on HP-UX IPF, Linux, and Oracle Solaris
- `LIBPATH` on AIX
- `SHLIB_PATH` on HP-UX PA-RISC

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

```
env
```

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

ODBCINI

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

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

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

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

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

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

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

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

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

See also

Data source configuration on UNIX/Linux on page 39

ODBCINST

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

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

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

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

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

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

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

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

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

See also

DSN-less connections on page 42

DD_INSTALLDIR

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

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

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

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

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

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

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

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

The next step is to test load the driver.

See also

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

The test loading tool

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

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

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

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

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

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

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

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

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

See also

Version string information on page 13

Data source configuration on UNIX/Linux

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

Configuration through the system information (odbc.ini) file

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

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

```
Amazon Redshift Wire Protocol=DataDirect 8.0 Amazon Redshift Wire Protocol
```

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, `[Redshift 2]`. The data source definitions contain connection string *`attribute`*`=`*`value`* pairs with default values. You can modify these values as appropriate for your system. "Connecton Option Descriptions" describes these attributes. See "Sample default odbc.ini file" for sample data sources.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 93
Sample default odbc.ini file on page 40
File data sources on page 44
IANAAppCodePage on page 119
DSN-less connections on page 42

Sample default odbc.ini file

The following is a sample `odbc.ini` file that Setup 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]
Amazon Redshift Wire Protocol=DataDirect 8.0 Amazon Redshift Wire Protocol

[Amazon Redshift Wire Protocol]
Driver=ODBCHOME/lib/ivrsft28.so
Description=DataDirect 8.0 Amazon Redshift Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
AutoCreate=0
AWSCluster=
AWSDBGroup=
AWSDBUser=
AWSRegion=
AzureClientID=
AzureClientSecret=
AzureTenantID=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
```

```
CryptoLibName=
CryptoProtocolVersion=TLSv1.2,TLSv1.1,TLSv1
Database=
DataSourceName=
EnableDescribeParam=1
EnableFIPS=1
EncryptionMethod=0
ExtendedColumnMetaData=0
ExtendedOptions=
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
HostName=
HostNameInCertificate=
InitializationString=
KeepAlive=0
KeyPassword=
Keystore=
KeystorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
LogonID=
LoginTimeout=15
MaxCharSize=
MaxPoolSize=100
MaxVarcharSize=
MinPoolSize=0
Password=
Pooling=0
PortNumber=5439
ProxyHost=
ProxyMode=0
ProxyPassword=
ProxyPort=0
ProxyUser=
QueryTimeout=0
ReportCodepageConversionErrors=0
ShowSelectableTables=1
SSLLibName=
Truststore=
TruststorePassword=
ValidateServerCertificate=1


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

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

- **To modify a data source:**
  a) Using a text editor, open the `odbc.ini` file.
  b) Modify the default attributes in the data source definitions as necessary based on your system specifics, for example, enter the host name and port number of your system in the appropriate location.

Consult the Amazon Redshift Wire Protocol Driver Attribute Names table in the Connection Options chapter 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 on page 93 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, `[Redshift]`.

d) Modify the attributes in the new definition as necessary based on your system specifics, for example, enter the host name and port number of your system in the appropriate location.

Consult the Amazon Redshift Wire Protocol Driver Attribute Names table in the Connection Options chapter 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 Amazon Redshift Wire Protocol Driver Attribute Names table in the Connection Options chapter 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.

The example application

Progress DataDirect ships an application, named *example*, that is installed in the `/samples/example` subdirectory of the product installation directory. Once you have configured your environment and data source, use the example application to test passing SQL statements. To run the application, enter `example` and follow the prompts to enter your data source name, user name, and password.

If successful, a SQL> prompt appears and you can type in SQL statements, such as `SELECT * FROM` *`table_name`*. If *example* is unable to connect to the database, an appropriate error message appears.

Refer to the `example.txt` file in the `example` subdirectory for an explanation of how to build and use this application.

Refer to "The example application" in *Progress DataDirect for ODBC Drivers Reference* for more information.

DSN-less connections

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

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

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

```
DataDirect 8.0 Amazon Redshift Wire Protocol=Installed.
```

This section also includes additional information for each driver.

The next section of the file is named `[Administrator]`. The keyword in this section, `AdminHelpRootDirectory`, is required for the Linux ODBC Administrator to locate its help system. The installation process automatically provides the correct value for this keyword.

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

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

See also

ODBCINST on page 37

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

Sample odbcinst.ini file

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

```
[ODBC Drivers]
DataDirect 8.0 Amazon Redshift Wire Protocol=Installed

[DataDirect 8.0 Amazon Redshift Wire Protocol]
Driver=ODBCHOME/lib/ivrsft28.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivrsft28.so
SQLLevel=0

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

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

File data sources

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

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

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

```
[ODBC]
Driver=DataDirect 8.0 Amazon Redshift Wire Protocol
Port=5439
HostName=Redshift2
LogonID=JOHN
```

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\Redshift2.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/Redshift2.dsn
```

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

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

```
FILEDSN=/home/users/john/filedsn/Redshift2.dsn;UID=james;PWD=test01
```

See also

Getting started on page 21

DSN-less connections on page 42

Data source configuration on UNIX/Linux on page 39

UTF-16 applications on UNIX and Linux

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

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

- The definition of SQLWCHAR in the ODBC header files must be switched from "char *" to "short *". To do this, the application uses #define SQLWCHARSHORT.
- The application must set the encoding for the environment or connection using one of the following attributes. If your application passes UTF-8 encoded strings to some connections and UTF-16 encoded strings to other

connections in the same environment, encoding should be set for the connection only; otherwise, either method can be used.

- To configure the encoding for the environment, set the ODBC environment attribute SQL_ATTR_APP_UNICODE_TYPE to a value of `SQL_DD_CP_UTF16`, for example:

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

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

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

Data source configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Redshift data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect for ODBC program group.
2. Select a tab:

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

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

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

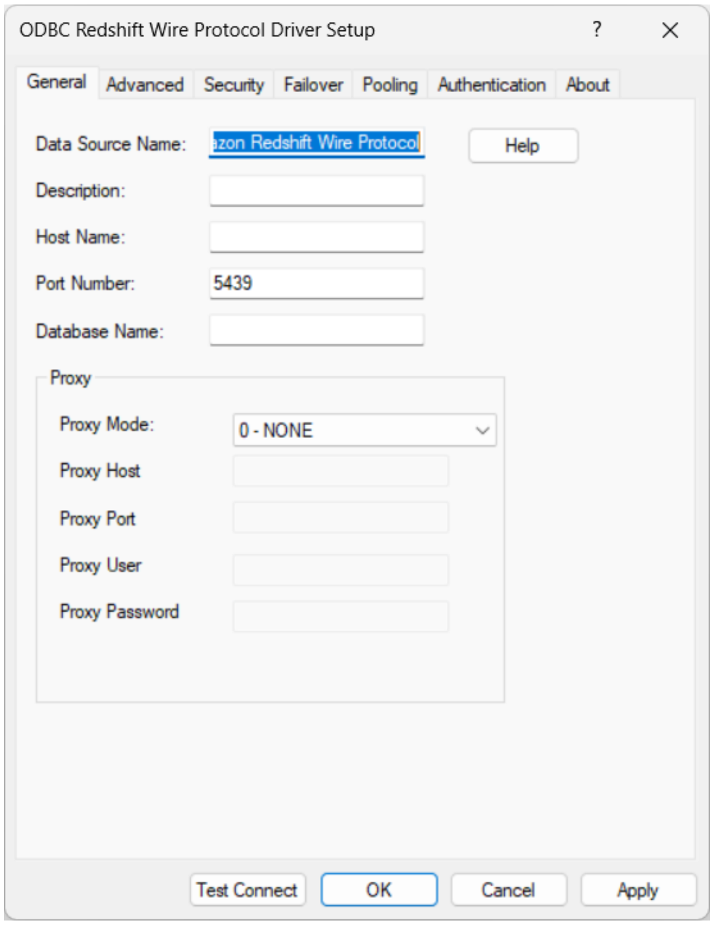
     If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click `Finish` to display the driver Setup dialog box.

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

     If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced**if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

3. The General tab of the Setup dialog box appears by default.

Figure 1: General tab



ODBC Redshift Wire Protocol Driver Setup

Data Source Name: azon Redshift Wire Protocol

Description:

Host Name:

Port Number: 5439

Database Name:

Proxy Mode: 0 - NONE

Proxy Host

Proxy Port

Proxy User

Proxy Password

On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options. The General tab displays fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

Connection Options: General	Description
Data Source Name on page 110	Specifies the name of a data source in your Windows Registry or `odbc.ini` file. **Default:** None
Description on page 111	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. **Default:** None
Host Name on page 117	The IP address endpoint of the Amazon Redshift cluster to which you want to connect. **Default:** None

Connection Options: General	Description
Port Number on page 129	Specifies the port number of the server listener. **Default:** `5439`
Database Name on page 110	Specifies the name of the database to which you want to connect. **Default:** None
Proxy Mode on page 130	Determines whether the driver connects to an Amazon Redshift endpoint through an HTTP proxy server. If set to **0 - NONE**, the driver connects directly to the Amazon Redshift endpoint specified by the Host Name connection option. If set to **1 - HTTP**, the driver connects to the Amazon Redshift endpoint through the HTTP proxy server specified by the ProxyHost connection option. **Default: 0 - None**
Proxy Host on page 129	Specifies the Hostname and possibly the Domain of the Proxy Server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address. **Default:** Empty string
Proxy Port on page 132	Specifies the port number where the Proxy Server is listening for HTTP requests. **Default:** `0`
Proxy User on page 133	Specifies the user name needed to connect to the Proxy Server. **Default:** Empty string
Proxy Password on page 131	Specifies the password needed to connect to the Proxy Server. **Default:** Empty string

4. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a logon dialog box" for details). Note that the information you enter in the logon dialog box during a test connect is not saved.
5. To further configure your driver, click on the following tabs. The corresponding sections provide details on the fields specific to each configuration tab:
   - Advanced tab allows you to configure advanced behavior.
   - Security tab allows you to specify security data source settings.
   - Failover tab allows you to specify failover data source settings.
   - Pooling tab allows you to specify connection pooling settings.

- Authentication tab allows you to specify authentication settings

6. Click **OK**. When you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

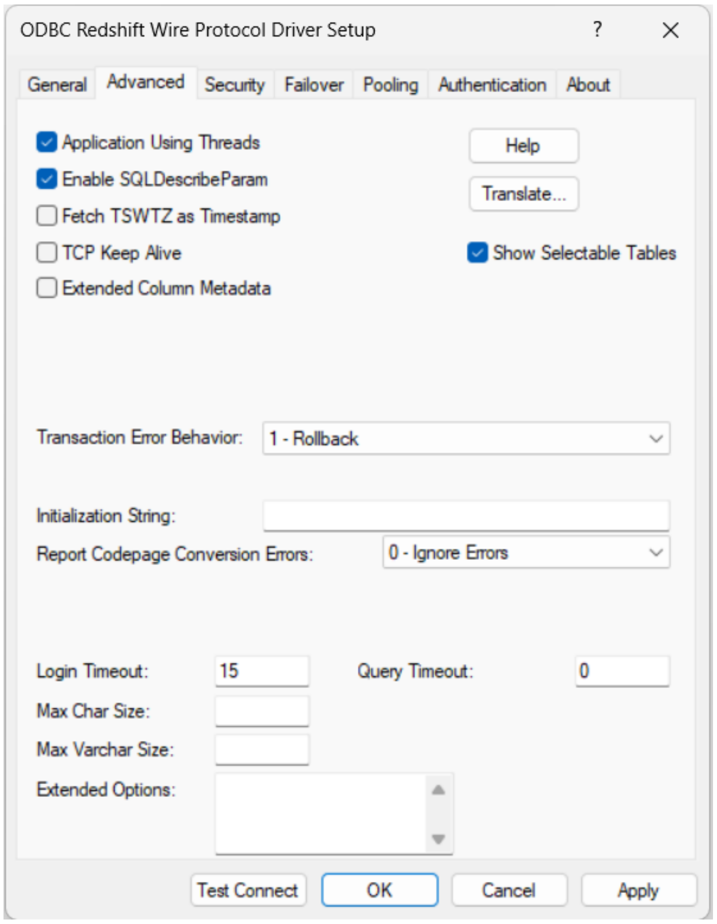
See also

Using a logon dialog box on page 64

Advanced tab

The Advanced tab allows you to specify additional data source settings. The fields are optional unless otherwise noted. On this tab, provide values for the options in the following table; then, click **Apply**.

Figure 2: Advanced tab



[✓] Application Using Threads
[✓] Enable SQLDescribeParam
[ ] Fetch TSWTZ as Timestamp
[ ] TCP Keep Alive
[ ] Extended Column Metadata
[✓] Show Selectable Tables

Transaction Error Behavior: 1 - Rollback

Initialization String:

Report Codepage Conversion Errors: 0 - Ignore Errors

Login Timeout: 15

Query Timeout: 0

Max Char Size:

Max Varchar Size:

Extended Options:

Connection Options: Advanced	Description
Application Using Threads on page 99	Determines whether the driver works with applications using multiple ODBC threads. If enabled, the driver works with single-threaded and multi-threaded applications. If disabled, the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards. **Default:** Enabled
Enable SQLDescribeParam on page 112	Determines whether SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement. If enabled, SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement. If disabled, the driver does not support SQLDescribeParam and returns the message: `Driver does not support this function.` **Default:** Enabled
Fetch TSWTZ as Timestamp on page 117	Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR. If enabled, the driver returns column values with the timestamp with time zone data type as the ODBC type SQL_TYPE_TIMESTAMP. The time zone information in the fetched value is truncated. Use this value if your application needs to process values the same way as TIMESTAMP columns. If disabled, the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_VARCHAR. Use this value if your application requires the time zone information in the fetched value. **Default:** Disabled
TCP Keep Alive on page 136	Specifies whether the driver enables TCPKeepAlive. If disabled, the driver does not enable TCPKeepAlive. If enabled, the driver enables TCPKeepAlive. **Default:** Disabled

Connection Options: Advanced	Description
Show Selectable Tables on page 135	Determines whether the driver returns metadata for all tables or only those for which the user has Select privileges. It can help the users with restricted Select privileges retrieve metadata for the required tables. If enabled, the driver returns metadata for all tables, irrespective of Select privileges. If disabled, the driver returns metadata only for the tables for which the user has Select privileges. Default: Enabled
Extended Column MetaData on page 114	Determines how the driver returns column metadata when using SQLDescribeCol and SQLColAttribute. For more information, see "Extended Column MetaData." **Default:** Disabled
Transaction Error Behavior on page 137	Determines how the driver handles errors that occur within a transaction. When an error occurs in a transaction, the Redshift server does not allow any operations on the connection except for rolling back the transaction. If set to **0 - Do Nothing**, the driver does not roll back the transaction when an error occurs. The application must handle the error and roll back the transaction. Any operation on the statement other than a rollback results in an error. If set to **1 - Rollback**, the driver rolls back the transaction when an error occurs. In addition to the original error message, the driver posts an error message indicating that the transaction has been rolled back. **Default: 1 - Rollback**
Initialization String on page 120	A SQL command that is issued immediately after connecting to the database to manage session settings. **Default:** None
Report Codepage Conversion Errors on page 134	Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another. If set to **0 - Ignore Errors**, the driver substitutes 0x1A for each character that cannot be converted and does not return a warning or error. If set to **1 - Return Error**, the driver returns an error instead of substituting 0x1A for unconverted characters. If set to **2 - Return Warning**, the driver substitutes 0x1A for each character that cannot be converted and returns a warning. **Default: 0 - Ignore Errors**

Connection Options: Advanced	Description
Login Timeout on page 124	The number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error. **Default:** `15`
Query Timeout on page 133	The number of seconds for the default query timeout for all statements that are created by a connection. **Default:** `0`
Max Char Size on page 125	Specifies the maximum size of columns of type SQL_CHAR that the driver describes through result set descriptions and catalog functions. **Default:** None
Max Varchar Size on page 126	Specifies the maximum size of columns of type SQL_VARCHAR that the driver describes through result set descriptions and catalog functions. **Default:** None

Extended Options: Type a semi-colon separated list of connection options and their values. Use this configuration option to set the value of undocumented connection options that are provided by Progress DataDirect Customer Support. You can include any valid connection option in the Extended Options string, for example:

```
Database=myredshift;UndocumentedOption1=value [;UndocumentedOption2=value;]
```

If the Extended Options string contains option values that are also set in the setup dialog or data source, the values of the options specified in the Extended Options string take precedence. However, connection options that are specified on a connection string override any option value specified in the Extended Options string.

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. Progress DataDirect provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

If you finished configuring your driver, proceed to Step 6 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Security tab allows you to specify security data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Authentication tab allows you to specify authentication settings.

See also

Extended Column MetaData on page 114

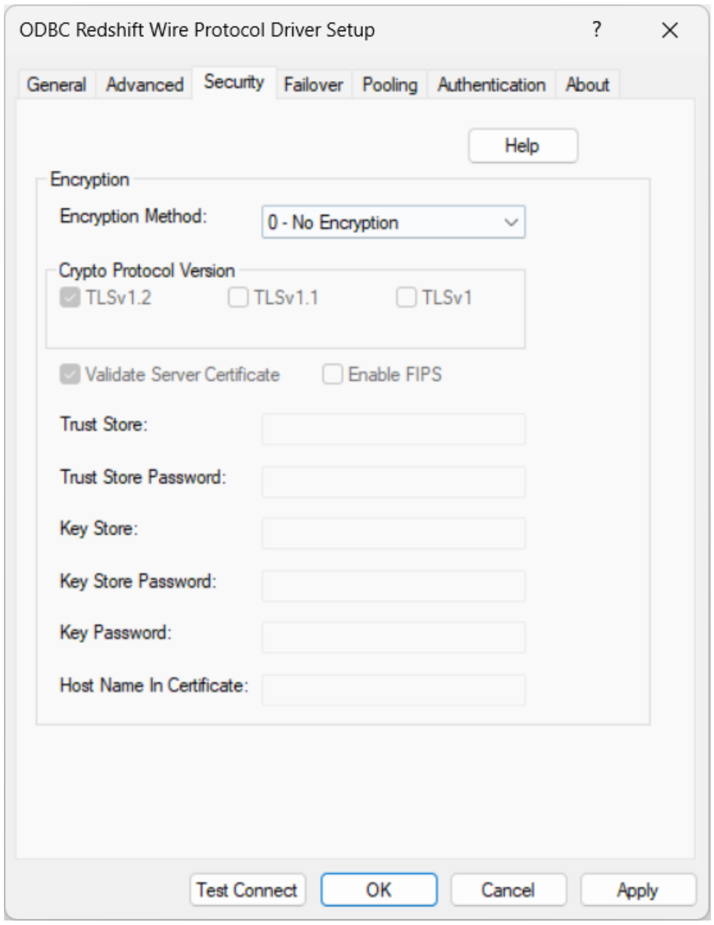
Data source configuration through a GUI on page 45

Security tab

The Security tab allows you to specify your security settings. The fields are optional unless otherwise noted. On this tab, provide values for the options in the following table; then, click **Apply**.

See "Using security" for a general description of authentication and encryption and their configuration requirements.

Figure 3: Security tab



Encryption Method: 0 - No Encryption

[✓] TLSv1.2 [ ] TLSv1.1 [ ] TLSv1

[✓] Validate Server Certificate [ ] Enable FIPS

Trust Store:

Trust Store Password:

Key Store:

Key Store Password:

Key Password:

Host Name In Certificate:

Connection Options: Security	Description
Encryption Method on page 113	The method the driver uses to encrypt data sent between the driver and the database server. If set to **0 - No Encryption**, data is not encrypted. If set to **1 - SSL**, data is encrypted using SSL. If the server is not configured for SSL, the connection fails. If set to **6 - RequestSSL**, the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. **Default: 0 - No Encryption**

Connection Options: Security	Description
Crypto Protocol Version on page 108	Specifies the cryptographic protocols to use when SSL is enabled using the Encryption Method connection option. **Default: TLSv1.2**, **TLSv1.1**, **TLSv1**
Validate Server Certificate on page 140	Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (`EncryptionMethod=1` | `6`). If enabled, the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. If disabled, the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options. **Default:** Enabled
Enable FIPS on page 111	Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled (`Encryption Method=1`). If disabled, the OpenSSL library uses cryptographic algorithms from the default provider. If enabled, the OpenSSL library uses cryptographic algorithms from the FIPS provider. **Default:** Disabled
Truststore on page 138	The directory that contains the truststore file and the truststore file name to be used when SSL is enabled (`EncryptionMethod=1` | `6`) and server authentication is used. **Default:** None
Truststore Password on page 138	The password that is used to access the truststore file when SSL is enabled (`EncryptionMethod=1` | `6`) and server authentication is used. **Default:** None
Key Store on page 121	The name of the directory containing the keystore file to be used when SSL is enabled (`Encryption Method=1` | `6`) and SSL client authentication is enabled on the database server. **Default:** None
Key Store Password on page 122	The password used to access the keystore file when SSL is enabled (`Encryption Method=1` | `6`) and SSL client authentication is enabled on the database server. **Default:** None

Connection Options: Security	Description
Key Password on page 120	The password used to access the individual keys in the keystore file when SSL is enabled (`EncryptionMethod=1` | `6`) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key. **Default:** None
Host Name In Certificate on page 118	A host name for certificate validation when SSL encryption is enabled (`EncryptionMethod=1` | `6`) and validation is enabled (`ValidateServerCertificate=1`). **Default:** None

If you finished configuring your driver, proceed to Step 6 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Authentication tab allows you to specify authentication settings.

See also

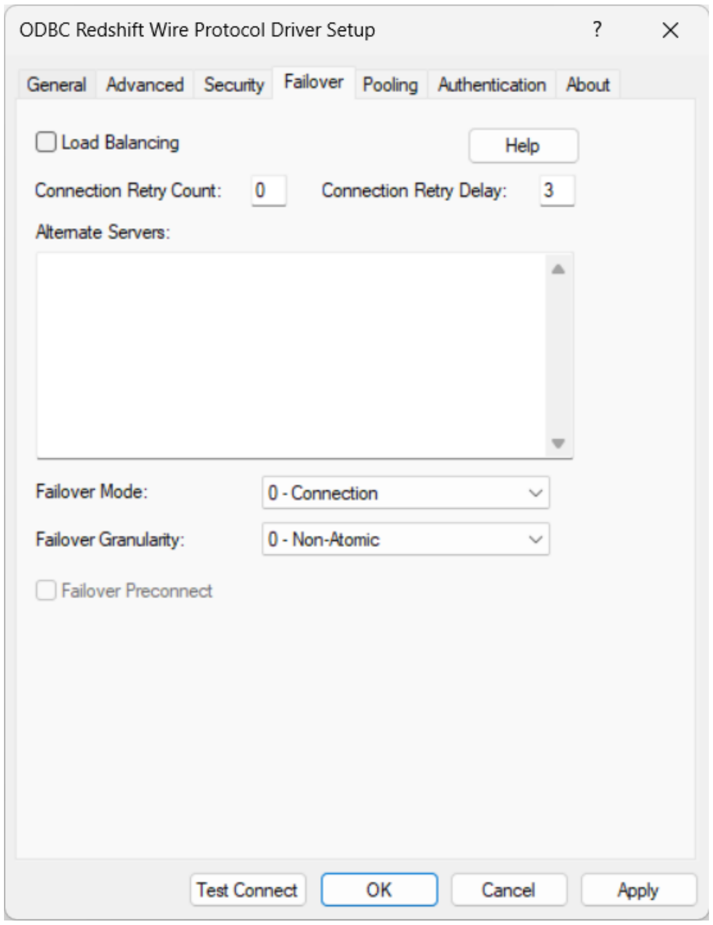
Using security on page 75

Data source configuration through a GUI on page 45

Failover tab

The Failover tab allows you to specify your failover datasource settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted. See "Using Failover" for a general description of failover and its related connection options.

Figure 4: Failover tab



Load Balancing: ☐
Connection Retry Count: 0
Connection Retry Delay: 3
Failover Mode: 0 - Connection
Failover Granularity: 0 - Non-Atomic
Failover Preconnect: ☐

Connection Options: Failover	Description
Load Balancing on page 123	Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). If enabled, the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order. If disabled, the driver does not use client load balancing and connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified). **Default:** Disabled
Connection Retry Count on page 106	The number of times the driver retries connection attempts to the primary database server, and if specified, alternate servers until a successful connection is established. **Default:** `0`

Connection Options: Failover	Description
Connection Retry Delay on page 107	Specifies the number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer. **Default:** 3
Alternate Servers on page 98	A list of alternate database servers to which the driver tries to connect if the primary database server is unavailable. Specifying a value for this option enables connection failover for the driver. The value you specify must be in the form of a string that defines the physical location of each alternate server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection. For additional information, see "Alternate Servers." **Default:** None
Failover Mode on page 115	Specifies the type of failover method the driver uses. If set to **0 - Connection**, the driver provides failover protection for new connections only. If set to **1 - Extended Connection**, the driver provides failover protection for new and lost connections, but not any work in progress. If set to **2 - Select**, the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed. **Default: 0 - Connection**

Connection Options: Failover	Description
Failover Granularity on page 115	Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection. If set to **0 - Non-Atomic**, the driver continues with the failover process and posts any errors on the statement on which they occur. If set to **1 - Atomic**, the driver fails the entire failover process if an error is generated as the result of anything other than executing and repositioning a Select statement. If an error is generated as a result of repositioning a result set to the last row position, the driver continues with the failover process, but generates a warning that the Select statement must be reissued. If set to **2 - Atomic Including Repositioning**, the driver fails the entire failover process if any error is generated as the result of restoring the state of the connection or the state of work in progress. If set to **3 - Disable Integrity Check**, the driver does not verify that the rows that were restored during the failover process match the original rows. This value applies only when Failover Mode is set to **2 - Select**. **Default: 0 - Non-Atomic**
Failover Preconnect on page 116	Specifies whether the driver tries to connect to the primary and an alternate server at the same time. If disabled, the driver tries to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection. If enabled, the driver tries to connect to the primary and an alternate server at the same time. This can be useful if your application is time-sensitive and cannot absorb the wait for the failover connection to succeed. **Default:** Disabled

If you finished configuring your driver, proceed to Step 6 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Authentication tab allows you to specify authentication settings.

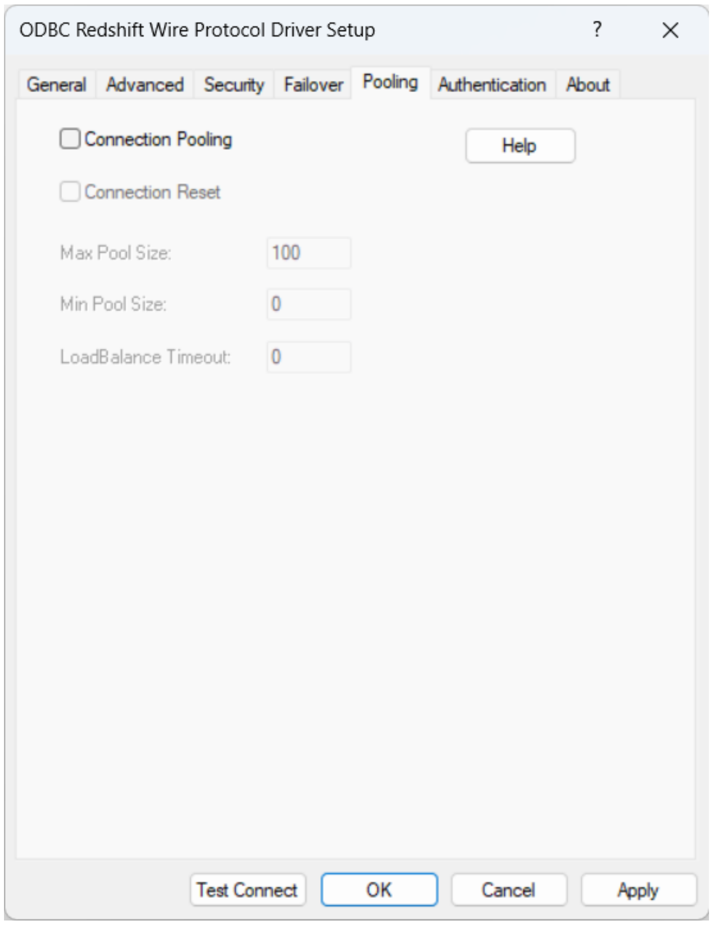
See also

Using failover on page 65
Alternate Servers on page 98
Data source configuration through a GUI on page 45

Pooling tab

The Pooling tab allows you to specify your pooling datasource settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted.

Figure 5: Pooling tab



Connection Pooling

Connection Reset

Max Pool Size: 100

Min Pool Size: 0

LoadBalance Timeout: 0

Connection Options: Pooling	Description
Connection Pooling on page 105	Specifies whether to use the driver's connection pooling. If enabled, the driver uses connection pooling. If disabled, the driver does not use connection pooling. **Default:** Disabled

Connection Options: Pooling	Description
Connection Reset on page 106	Determines whether the state of connections that are removed from the connection pool for reuse by the application is reset to the initial configuration of the connection. If enabled, the state of connections removed from the connection pool for reuse by an application is reset to the initial configuration of the connection. Resetting the state can negatively impact performance because additional commands must be sent over the network to the server to reset the state of the connection. If disabled, the state of connections is not reset. **Default:** Disabled
Max Pool Size on page 125	The maximum number of connections allowed within a single connection pool. When the maximum number of connections is reached, no additional connections can be created in the connection pool. **Default:** `100`
Min Pool Size on page 126	The minimum number of connections that are opened and placed in a connection pool, in addition to the active connection, when the pool is created. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value. **Default:** `0`
Load Balance Timeout on page 122	Specifies the number of seconds to keep inactive connections open in a connection pool. An inactive connection is a database session that is not associated with an ODBC connection handle, that is, a connection in the pool that is not in use by an application. **Default:** `0`

If you finished configuring your driver, proceed to Step 6 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security allows you to specify security data source settings.
- Failover tab allows you to specify failover data source settings.
- Authentication tab allows you to specify authentication settings.

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

See also

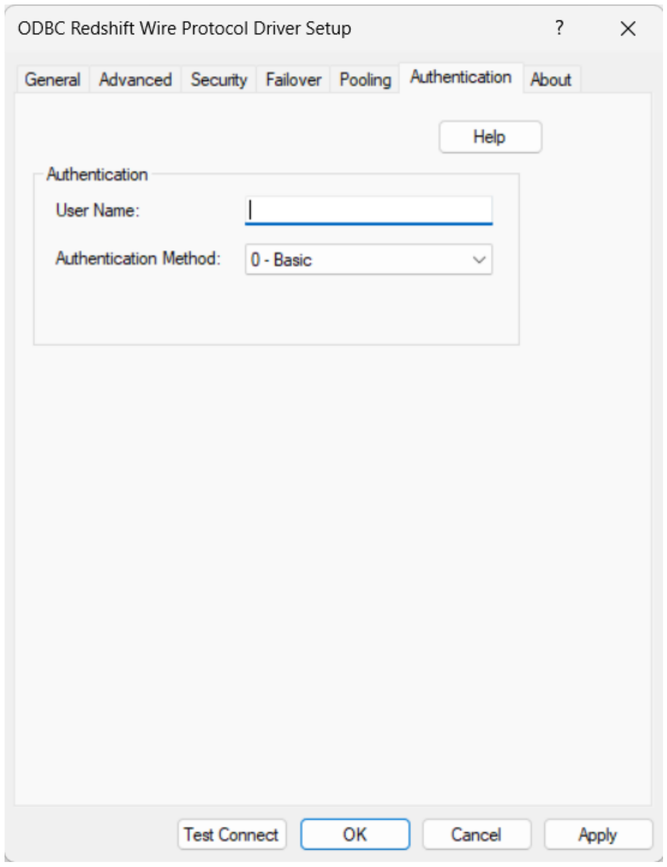
Data source configuration through a GUI on page 45

Authentication tab

The Authentication tab allows you to specify your authentication settings. On this tab, select the authentication method you want to use to connect to the database and provide values for the relevant options; then, click **Apply**.

See "Authentication" for a general description and configuration details of the authentication methods supported by the driver.

Figure 6: Authentication tab



User Name:

Authentication Method: 0 - Basic

Connection Options: Security	Description
User Name on page 139	The default user ID that is used to connect to your database. Your ODBC application may override this value or you may override it in the logon dialog box or connection string. **Default:** None

Connection Options: Security	Description
Authentication Method on page 99	Determines which authentication method the driver uses when establishing a connection. If set to **0 - Basic**, the driver uses a hashed value, based on the concatenation of the user name and password, for authentication. To specify the values required for Basic authentication, click **Test Connect**. If set to **13 - AzureAD**, the driver uses Azure AD authentication when establishing a connection. **Default: 0 - Basic**
Azure Client ID on page 103	Specifies the client ID key for your application when authenticating to Amazon Redshift using Azure AD authentication. **Default:** None
Azure Client Secret on page 104	Specifies the client secret for your application when authenticating to Amazon Redshift using Azure AD authentication. **Default:** None
Azure Tenant ID on page 104	Specifies the ID of the Azure AD instance in which you create and manage your application. The driver uses this value when authenticating to Amazon Redshift using Azure AD authentication. **Default:** None
AWS DB User on page 102	Specifies your AWS user name when authenticating to Amazon Redshift using Azure AD authentication. **Default:** None
Auto Create on page 100	Determines whether the driver creates a new AWS user when authenticating to Amazon Redshift using Azure AD authentication. **Note:** When creating a new AWS user, you must specify the name for the new AWS user in the AWS DB User field. **Default:** Disabled
AWS DB Group on page 101	Specifies the name of the AWS user group that your AWS user name is a part of. The driver uses this value when authenticating to Amazon Redshift using Azure AD authentication. This is an optional field. **Default:** None

Connection Options: Security	Description
AWS Region on page 103	Specifies the name of the region that hosts your AWS server. The driver uses this value when connecting to an Amazon Redshift database using Azure AD authentication. **Note:** The values must be specified for either AWS Region and AWS Cluster or Host Name and Port Number. If they are specified for all four of them, the values for Host Name and Port Number take precedence over those for AWS Region and AWS Cluster. **Default:** None
AWS Cluster on page 101	Specifies the name of the Amazon Redshift cluster that contains the database you want to connect to. The driver uses this value when connecting to an Amazon Redshift database using Azure AD authentication. **Note:** The values must be specified for either AWS Region and AWS Cluster or Host Name and Port Number. If they are specified for all four of them, the values for Host Name and Port Number take precedence over those for AWS Region and AWS Cluster. **Default:** None

If you finished configuring your driver, proceed to Step 4 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.

See also

Authentication on page 85
Data source configuration through a GUI on page 45

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 Amazon Redshift for Linux/UNIX/Windows is:

```
DSN=Redshift;UID=JOHN;PWD=XYZZY
```

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

```
FILEDSN=Redshift;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire
Protocol;HOST=RedshiftServer;PORT=5439;UID=JOHN;PWD=XYZZY;DB=REDSHIFT2
```

See also

Connection option descriptions on page 93

Password Encryption Tool (UNIX/Linux only)

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

- KeyPassword
- KeyStorePassword
- TrustStorePassword
- Password

To use the Password Encryption Tool:

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

   ```
   ddencpwd password
   ```

   where:

`password`

is the password you want to encrypt.

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

   `KeyPassword=`*`returned_value`*

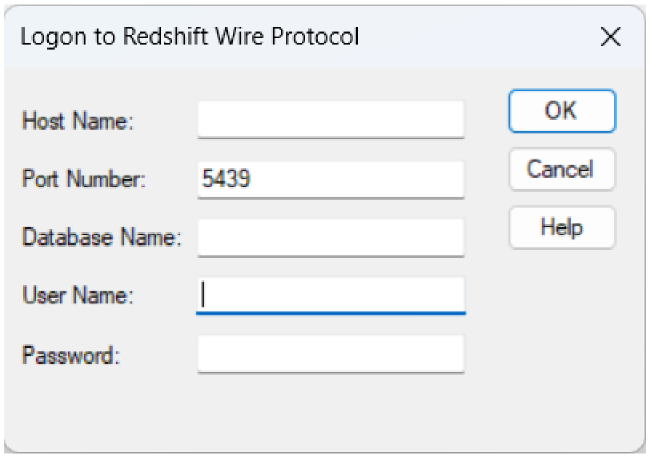
4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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

Using a logon dialog box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the host name has already been specified.

Figure 7: Logon to Amazon Redshift dialog box



Host Name:
Port Number: 5439
Database Name:
User Name:
Password:

In this dialog box, provide the following information:

1. In the Host Name field, type the name or the IP address of the server to which you want to connect.
2. In the Port Number field, type the port number of the server listener.
3. In the Database Name field, type the name of the database to which you want connect.
4. In the User Name field, type your logon ID.
5. In the Password field, type your password.
6. Click **OK** to complete the logon.

Performance considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design.

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

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

Connection Pooling (Pooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout (LoadBalanceTimeout)**: You can define how long to keep connections in the pool. The time that a connection was last used is compared to the current time and, if the timespan exceeds the value of the Load Balance Timeout option, the connection is destroyed. The Min Pool Size option can cause some connections to ignore this value.
- **Connection Reset (ConnectionReset)**: Resetting a re-used connection to the initial configuration settings impacts performance negatively because the connection must issue additional commands to the server.
- **Max Pool Size (MaxPoolSize)**: Setting the maximum number of connections that the pool can contain too low might cause delays while waiting for a connection to become available. Setting the number too high wastes resources.
- **Min Pool Size (MinPoolSize)**: A connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size, if one has been specified. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

Encryption Method (**EncryptionMethod**): Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) that is required to encrypt and decrypt data.

Failover Mode (FailoverMode): Although high availability that replays queries after a failure provides increased levels of protection, it can adversely affect performance because of increased overhead.

Using failover

To ensure continuous, uninterrupted access to data, the Progress DataDirect *for* ODBC driver provides the following levels of failover protection, listed from basic to more comprehensive:

- *Connection failover* provides failover protection for new connections only. The driver fails over new connections to an alternate, or backup, database server if the primary database server is unavailable, for example, because of a hardware failure or traffic overload. If a connection to the database is lost, or dropped, the driver does not fail over the connection. This failover method is the default.
- *Extended connection failover* provides failover protection for new connections and lost database connections. If a connection to the database is lost, the driver fails over the connection to an alternate server, preserving the state of the connection at the time it was lost, but not any work in progress.
- *Select Connection failover* provides failover protection for new connections and lost database connections. In addition, it provides protection for Select statements that have work in progress. If a connection to the database is lost, the driver fails over the connection to an alternate server, preserving the state of the connection at the time it was lost and preserving the state of any work being performed by Select statements.

The method you choose depends on how failure tolerant your application is. For example, if a communication failure occurs while processing, can your application handle the recovery of transactions and restart them? Your application needs the ability to recover and restart transactions when using either extended connection failover mode or select connection failover mode. The advantage of select mode is that it preserves the state of any work that was being performed by the Select statement at the time of connection loss. If your application had been iterating through results at the time of the failure, when the connection is reestablished the driver can reposition on the same row where it stopped so that the application does not have to undo all of its previous result processing. For example, if your application were paging through a list of items on a Web page when a failover occurred, the next page operation would be seamless instead of starting from the beginning. Performance, however, is a factor in selecting a failover mode. Select mode incurs additional overhead when tracking what rows the application has already processed.

You can specify which failover method you want to use by setting the "Failover Mode" connection option. Read the following sections for details on each failover method:

- Connection failover on page 66
- Extended connection failover on page 67
- Select connection failover on page 69

Regardless of the failover method you choose, you must configure one or multiple alternate servers using the "Alternate Servers" connection option. See "Guidelines for primary and alternate servers" for information about primary and alternate servers.

See also

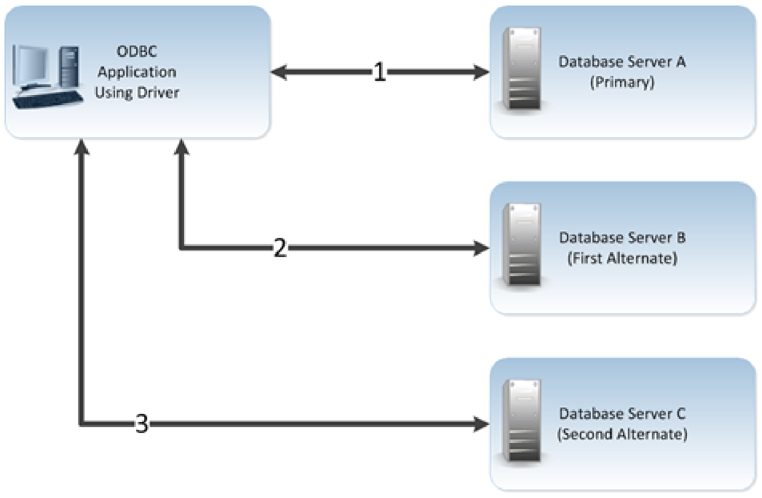
Failover Mode on page 115
Alternate Servers on page 98
Guidelines for primary and alternate servers on page 70

Connection failover

Connection failover allows an application to connect to an alternate, or backup, database server if the primary database server is unavailable, for example, because of a hardware failure or traffic overload. Connection failover provides failover protection for new connections only and does not provide protection for lost connections to the database, nor does it preserve states for transactions or queries.

You can customize the drivers for connection failover by configuring a list of alternate database servers that are tried if the primary server is not accepting connections. Connection attempts continue until a connection is successfully established or until all the alternate database servers have been tried the specified number of times.

For example, suppose you have the environment shown in the following illustration with multiple database servers: Database Server A, B, and C. Database Server A is designated as the primary database server, Database Server B is the first alternate server, and Database Server C is the second alternate server.


ODBC Application Using Driver
1
2
3
Database Server A (Primary)
Database Server B (First Alternate)
Database Server C (Second Alternate)

First, the application attempts to connect to the primary database server, Database Server A (**1**). If connection failover is enabled and Database Server A fails to accept the connection, the application attempts to connect to Database Server B (**2**). If that connection attempt also fails, the application attempts to connect to Database Server C (**3**).

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the driver can retry each alternate database server (primary and alternate) for a specified number of attempts. You can specify the number of attempts that are made through the *connection retry* feature. You can also specify the number of seconds of delay, if any, between attempts through the *connection delay* feature. See "Using connection retry" for more information about connection retry.

A driver fails over to the next alternate database server only if a successful connection cannot be established with the current alternate server. If the driver successfully establishes communication with a database server and the connection request is rejected by the database server because, for example, the login information is invalid, then the driver generates an error and does not try to connect to the next database server in the list. It is assumed that each alternate server is a mirror of the primary and that all authentication parameters and other related information are the same.

For details on configuring connection failover for your driver, see "Configuring failover-related options."

See also

Using connection retry on page 71

Configuring failover-related options on page 72

Extended connection failover

Extended connection failover provides failover protection for the following types of connections:

- New connections, in the same way as described in "Connection Failover"
- Lost connections

When a connection to the database is lost, the driver fails over the connection to an alternate server, restoring the same state of the connection at the time it was lost. For example, when reestablishing a lost connection on the alternate database server, the driver performs the following actions:

- Restores the connection using the same connection options specified by the lost connection
- Reallocates statement handles and attributes

- Logs in the user to the database with the same user credentials
- Restores any prepared statements associated with the connection and repopulates the statement pool
- Restores manual commit mode if the connection was in manual commit mode at the time of the failover

The driver does not preserve work in progress. For example, if the database server experienced a hardware failure while processing a query, partial rows processed by the database and returned to the client would be lost. If the driver was in manual commit mode and one or more Inserts or Updates were performed in the current transaction before the failover occurred, then the transaction on the primary server is rolled back. The Inserts or Updates done before the failover are not committed to the primary server. Your application needs to rerun the transaction after the failover because the Inserts or Updates done before the failover are not repeated by the driver on the failover connection.

When a failover occurs, if a statement is in allocated or prepared state, the next operation on the statement returns a SQL state of 01000 and a vendor code of 0. If a statement is in an executed or prepared state, the next operation returns a SQL state of 40001 and a vendor code of 0. Either condition returns an error message similar to:

```
Your connection has been terminated. However, you have been successfully connected to
the next available AlternateServer: 'HOSTNAME=Server4:PORTNUMBER= 1521:SERVICENAME=test'.
 All active transactions have been rolled back.
```

The driver retains all connection settings made through ODBC API calls when a failover connection is made. It does not, however, retain any session settings established through SQL statements. This can be done through the Initialization String connection option, described in the individual driver chapters.

The driver retains the contents of parameter buffers, which can be important when failing over after a fetch. All Select statements are re-prepared at the time the failover connection is made. All other statements are placed in an allocated state.

If an error occurs while the driver is reestablishing a lost connection, the driver can fail the entire failover process or proceed with the process as far as it can. For example, suppose an error occurred while reestablishing the connection because a table for which the driver had a prepared statement did not exist on the alternate connection. In this case, you may want the driver to notify your application of the error and proceed with the failover process. You can choose how you want the driver to behave if errors occur during failover by setting the Failover Granularity connection option.

During the failover process, your application may experience a short pause while the driver establishes a connection on an alternate server. If your application is time-sensitive (a real-time customer order application, for example) and cannot absorb this wait, you can set the Failover Preconnect connection option to true. Setting the Failover Preconnect option to true instructs the driver to establish connections to the primary server and an alternate server at the same time. Your application uses the first connection that is successfully established. If this connection to the database is lost at a later time, the driver saves time in reestablishing the connection on the server to which it fails over because it can use the spare connection in its failover process.

This pre-established failover connection is not used by the driver until the driver determines that it needs to fail over. If the server to which the driver is connected or the network equipment through which the connection is routed is configured with a timeout, the pre-configured failover connection could time out. The pre-configured failover connection can also be lost if the failover server is brought down and back up again. The driver tries to establish the connection to the failover server again if the connection is lost.

See also

Connection failover on page 66
Failover Granularity on page 115
Failover Preconnect on page 116

Select connection failover

Select connection failover provides failover protection for the following types of connections:

- New connections, in the same way as described in "Connection Failover"
- Lost connections, in the same way as described in "Extended Connection Failover"

Select connection failover provides failover protection for the following types of connections:

- New connections, in the same way as described in "Connection failover"
- Lost connections, in the same way as described in "Extended connection failover"

In addition, the driver can recover work in progress because it keeps track of the last Select statement the application executed on each Statement handle, including how many rows were fetched to the client. For example, if the database had only processed 500 of 1,000 rows requested by a Select statement when the connection was lost, the driver would reestablish the connection to an alternate server, re-execute the Select statement, and position the cursor on the next row so that the driver can continue fetching the balance of rows as if nothing had happened.

Performance, however, is a factor when considering whether to use Select mode. Select mode incurs additional overhead when tracking what rows the application has already processed.

The driver only recovers work requested by Select statements. You must explicitly restart the following types of statements after a failover occurs:

- Insert, Update, or Delete statements
- Statements that modify the connection state, for example, SET or ALTER SESSION statements
- Objects stored in a temporary tablespace or global temporary table
- Partially executed stored procedures and batch statements

When in manual transaction mode, no statements are rerun if any of the operations in the transaction were Insert, Update, or Delete. This is true even if the statement in process at the time of failover was a Select statement.

By default, the driver verifies that the rows that are restored match the rows that were originally fetched and, if they do not match, generates an error warning your application that the Select statement must be reissued. By setting the Failover Granularity connection option, you can customize the driver to ignore this check altogether or fail the entire failover process if the rows do not match.

When the row comparison does not agree, the default behavior of Failover Granularity returns a SQL state of 40003 and an error message similar to:

```
Unable to position to the correct row after a successful failover attempt to
AlternateServer: 'HOSTNAME=Server4:PORTNUMBER= 1521:SERVICENAME=test'. You must reissue
 the select statement.
```

If you have configured Failover Granularity to fail the entire failover process, the driver returns a SQL state of 08S01 and an error message similar to:

```
Your connection has been terminated and attempts to complete the failover process to the
 following Alternate Servers have failed: AlternateServer: 'HOSTNAME=Server4:PORTNUMBER=
 1521:SERVICENAME=test'. All active transactions have been rolled back.
```

When the row comparison does not agree, the default behavior of Failover Granularity returns a SQL state of 40003 and an error message similar to:

```
Unable to position to the correct row after a successful failover attempt to
AlternateServer: 'HOSTNAME=Server4:PORTNUMBER= 1521:SERVICENAME=test'. You must reissue
 the select statement.
```

If you have configured Failover Granularity to fail the entire failover process, the driver returns a SQL state of 08S01 and an error message similar to:

```
Your connection has been terminated and attempts to complete the failover process to the
 following Alternate Servers have failed: AlternateServer: 'HOSTNAME=Server4:PORTNUMBER=
 1521:SERVICENAME=test'. All active transactions have been rolled back.
```

See also

Connection failover on page 66

Extended connection failover on page 67

Guidelines for primary and alternate servers

For Amazon Redshift, the driver must connect to the leader node. Currently, Amazon Redshift only supports the notion of a single leader node. However, you can take advantage of the driver's advanced failover capabilities by setting the Alternate Servers connection option to the same host name or port number specified for the primary server in the Host Name and Port Number connection options. See "Alternate Servers" and "Port Number" for more information.

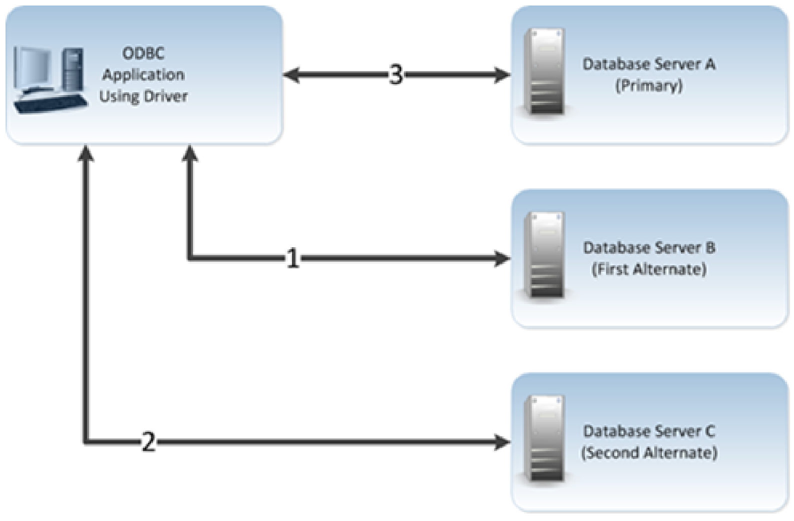
See also

Alternate Servers on page 98

Port Number on page 129

Using client load balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When client load balancing is enabled, the order in which primary and alternate database servers are tried is random. For example, suppose that client load balancing is enabled as shown in the following illustration:


ODBC Application Using Driver
Database Server A (Primary)
Database Server B (First Alternate)
Database Server C (Second Alternate)
1
2
3

First, Database Server B is tried (**1**). Then, Database Server C may be tried (**2**), followed by a connection attempt to Database Server A (**3**). In contrast, if client load balancing were not enabled in this scenario, each database server would be tried in sequential order, primary server first, then each alternate server based on its entry order in the alternate servers list.

Client load balancing is controlled by the "Load Balancing" connection option. For details on configuring client load balancing, see "Configuring and connecting to data sources."

See also

Load Balancing on page 123

Configuring and connecting to data sources on page 35

Using connection retry

Connection retry defines the number of times the driver attempts to connect to the primary server and, if configured, alternate database servers after the initial unsuccessful connection attempt. It can be used with connection failover, extended connection failover, and select failover. Connection retry can be an important strategy for system recovery. For example, suppose you have a power failure in which both the client and the server fails. When the power is restored and all computers are restarted, the client may be ready to attempt a connection before the server has completed its startup routines. If connection retry is enabled, the client application can continue to retry the connection until a connection is successfully accepted by the server.

Connection retry can be used in environments that have only one server or can be used as a complementary feature with connection failover in environments with multiple servers.

Using the connection options Connection Retry Count and Connection Retry Delay, you can specify the number of times the driver attempts to connect and the time in seconds between connection attempts. For details on configuring connection retry, see "Configuring Failover-Related Options."

See also

Connection Retry Count on page 106

Connection Retry Delay on page 107

Configuring failover-related options on page 72

Configuring failover-related options

The following table summarizes how failover-related connection options work with the driver. The connection options are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name is listed immediately after the GUI name in parentheses. See "Connection option descriptions" for details about configuring the options. The step numbers in the table refer the procedure that follows the table

Table 2: Summary: Failover and Related Connection Options

Option	Characteristic
Alternate Servers (AlternateServers) (See step 1 on page 73)	A list of alternate database servers to which the driver tries to connect if the primary database server is unavailable. Specifying a value for this option enables connection failover for the driver. The value you specify must be in the form of a string that defines the physical location of each alternate server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection. For additional information, see "Alternate Servers." **Default:** None
Connection Retry Count (ConnectionRetryCount) (See step 5 on page 74)	The number of times the driver retries connection attempts to the primary database server, and if specified, alternate servers until a successful connection is established. **Default:** `0`
Connection Retry Delay (ConnectionRetryDelay) (See step 6 on page 74)	Specifies the number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer. **Default:** `3`
Failover Granularity (FailoverGranularity) (See step 3 on page 73)	Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection. If set to `0` (Non-Atomic), the driver continues with the failover process and posts any errors on the statement on which they occur. If set to `1` (Atomic), the driver fails the entire failover process if an error is generated as the result of anything other than executing and repositioning a Select statement. If an error is generated as a result of repositioning a result set to the last row position, the driver continues with the failover process, but generates a warning that the Select statement must be reissued. If set to `2` (Atomic Including Repositioning), the driver fails the entire failover process if any error is generated as the result of restoring the state of the connection or the state of work in progress. If set to `3` (Disable Integrity Check), the driver does not verify that the rows that were restored during the failover process match the original rows. This value applies only when Failover Mode is set to **2 - Select**. **Default:** `0` (Non-Atomic)

Option	Characteristic
Failover Mode (FailoverMode) (See step 2 on page 73)	Specifies the type of failover method the driver uses. If set to `0` (Connection), the driver provides failover protection for new connections only. If set to `1` (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress. If set to `2` (Select), the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed. **Default:** `0` (Connection)
Failover Preconnect (FailoverPreconnect) (See step 4 on page 74)	Specifies whether the driver tries to connect to the primary and an alternate server at the same time. If disabled, the driver tries to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection. If enabled, the driver tries to connect to the primary and an alternate server at the same time. This can be useful if your application is time-sensitive and cannot absorb the wait for the failover connection to succeed. **Default:** `0` (Disabled)
Load Balancing (LoadBalancing) (See step 7 on page 74)	Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). If enabled, the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order. If disabled, the driver does not use client load balancing and connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified). **Default:** `0` (Disabled)

1. To configure connection failover, you **must** specify one or more alternate database servers that are tried at connection time if the primary server is not accepting connections. To do this, use the Alternate Servers connection option. Connection attempts continue until a connection is successfully established or until all the database servers in the list have been tried once (the default).
2. Choose a failover method by setting the Failover Mode connection option. The default method is Connection (`FailoverMode=0`).
3. If Failover Mode is Extended Connection (`FailoverMode=1`) or Select (FailoverMode=2), set the Failover Granularity connection option to specify how you want the driver to behave if errors occur while trying to reestablish a lost connection. The default behavior of the driver is Non-Atomic (`FailoverGranularity=0`), which continues with the failover process and posts any errors on the statement on which they occur. Other values are:

   Atomic (`FailoverGranularity=1`): the driver fails the entire failover process if an error is generated as the result of anything other than executing and repositioning a Select statement. If an error is generated as a result of repositioning a result set to the last row position, the driver continues with the failover process, but generates a warning that the Select statement must be reissued.

   Atomic including Repositioning (`FailoverGranularity=2`): the driver fails the entire failover process if any error is generated as the result of restoring the state of the connection or the state of work in progress.

Disable Integrity Check (`FailoverGranularity=3`): the driver does not verify that the rows restored during the failover process match the original rows. This value applies only when Failover Mode is set to Select (`FailoverMode=2`).

4. Optionally, enable the Failover Preconnect connection option (`FailoverPreconnect=1`) if you want the driver to establish a connection with the primary and an alternate server at the same time. This value applies only when Failover Mode is set to Extended Connection (`FailoverMode=1`) or Select (`FailoverMode=2`). The default behavior is to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection (`FailoverPreconnect=0`).
5. Optionally, specify the number of times the driver attempts to connect to the primary and alternate database servers after the initial unsuccessful connection attempt. By default, the driver does not retry. To set this feature, use the Connection Retry Count connection option.
6. Optionally, specify the wait interval, in seconds, between attempts to connect to the primary and alternate database servers. The default interval is 3 seconds. To set this feature, use the Connection Retry Delay connection option.
7. Optionally, specify whether the driver will use client load balancing in its attempts to connect to primary and alternate database servers. If load balancing is enabled, the driver uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load balancing. To set this feature, use the Load Balancing connection option.

See also

Connection option descriptions on page 93

A connection string example

The following connection string configures the driver to use connection failover in conjunction with some of its optional features.

```
DSN=MyRedshiftDSN;AlternateServers=(HostName=RedshiftServer:PortNumber=5439:
Database=Redshift1, HostName=255.201.11.24:PortNumber=1522:Database=Redshift2);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0
```

Specifically, this connection string configures the driver to use two alternate servers as connection failover servers, to attempt to connect four additional times if the initial attempt fails, to wait five seconds between attempts, to try the primary and alternate servers in a random order, and to attempt reconnecting on new connections only. The additional connection information required for the alternate servers is specified in the data source RedshiftServer.

An odbc.ini File example

To configure the 32-bit driver to use connection failover in conjunction with some of its optional features in your `odbc.ini` file, you could set the following connection string attributes:

```
Driver=ODBCHOME/lib/ivrsftxx.so
Description=DataDirect 8.0 Amazon Redshift Wire Protocol
...
AlternateServers=(HostName=RedshiftServer:PortNumber=5439:Database=Redshift1,
HostName=255.201.11.24:PortNumber=1522:Database=Redshift2)
...
ConnectionRetryCount=4
ConnectionRetryDelay=5
...
LoadBalancing=0
...
FailoverMode=1
...
FailoverPreconnect=1
...
```

Specifically, this `odbc.ini` configuration tells the driver to use two alternate servers as connection failover servers, to attempt to connect four additional times if the initial attempt fails, to wait five seconds between attempts, to try the primary and alternate servers in sequential order (do not use load balancing), to attempt reconnecting on new and lost connections, and to establish a connection with the primary and alternate servers at the same time.

Using security

The driver supports the *data encryption* security feature, which converts data into a form that cannot be easily understood by unauthorized users.

Data encryption across the network

If your database connection is not configured to use data encryption, data is sent across the network in a format that is designed for fast transmission and can be decoded by interceptors, given some time and effort. For example, text data is often sent across the wire as clear text. Because this format does not provide complete protection from interceptors, you may want to use data encryption to provide a more secure transmission of data.

For example, you may want to use data encryption in the following scenarios:

- You have offices that share confidential information over an intranet.
- You send sensitive data, such as credit card numbers, over a database connection.
- You need to comply with government or industry privacy and security requirements.

Your Progress DataDirect *for* ODBC driver supports Transport Layer Security (TLS) and Secure Sockets Layer (SSL). TLS/SSL are industry-standard protocols for sending encrypted data over database connections. TLS/SSL secures the integrity of your data by encrypting information and providing client/server authentication.

Note: Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) required to encrypt and decrypt data.

TLS/SSL encryption

TLS/SSL works by allowing the client and server to send each other encrypted data that only they can decrypt. TLS/SSL negotiates the terms of the encryption in a sequence of events known as the *handshake*. During the handshake, the driver negotiates the highest TLS/SSL protocol available. The result of this negotiation determines the encryption cipher suite to be used for the TLS/SSL session.

The encryption cipher suite defines the type of encryption that is used for any data exchanged through a TLS/SSL connection. Some cipher suites are very secure and, therefore, require more time and resources to encrypt and decrypt data, while others provide less security, but are also less resource intensive.

The handshake involves the following types of authentication:

- *TLS/SSL server authentication* requires the server to authenticate itself to the client.
- *TLS/SSL client authentication* is optional and requires the client to authenticate itself to the server after the server has authenticated itself to the client.

Refer to "SSL encryption cipher suites" in the *Progress DataDirect for ODBC Drivers Reference* for a list of the encryption cipher suites supported by the drivers.

Certificates

TLS/SSL encryption requires the use of a digitally-signed document, an x.509 standard certificate, for authentication and the secure exchange of data. The purpose of this certificate is to tie the public key contained in the certificate securely to the person/company that holds the corresponding private key. Your Progress DataDirect *for* ODBC drivers supports many popular formats. Supported formats include:

- DER Encoded Binary X.509
- Base64 Encoded X.509
- PKCS #12 / Personal Information Exchange

TLS/SSL server authentication

When the client makes a connection request, the server presents its public certificate for the client to accept or deny. The client checks the issuer of the certificate against a list of trusted Certificate Authorities (CAs) that resides in an encrypted file on the client known as a *truststore*. If the certificate matches a trusted CA in the truststore, an encrypted connection is established between the client and server. If the certificate does not match, the connection fails and the driver generates an error.

Most truststores are password-protected. The driver must be able to locate the truststore and unlock the truststore with the appropriate password. Two connection options are available to the driver to provide this information: Trust Store (Truststore) and Trust Store Password (TruststorePassword). The value of Trust Store is a pathname that specifies the location of the truststore file. The value of Trust Store Password is the password required to access the contents of the truststore.

Alternatively, you can configure the driver to trust any certificate sent by the server, even if the issuer is not a trusted CA. Allowing a driver to trust any certificate sent from the server is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment. Setting the Validate Server Certificate (ValidateServerCertificate) connection option to false allows the driver to accept any certificate returned from the server regardless of whether the issuer of the certificate is a trusted CA.

Finally, the connection option, Host Name In Certificate (HostNameInCertificate), allows an additional method of server verification. When a value is specified for Host Name In Certificate, it must match the common name of the host in the Subject of the certificate. This prevents malicious intervention between the client and the server and ensures that the driver is connecting to the server that was requested.

To configure the driver to use data encryption via TLS/SSL server authentication:

- Set the Host Name (HostName) option to specify the name or the IP address of the server to which you want to connect.
- Set the Port Number (PortNumber) option to specify the port number of the server listener. The default is `5439`.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Trust Store (Truststore) option to specify either the full path of the truststore file or the contents of the TLS/SSL certificates.
- Set the Truststore Password (TruststorePassword) option to specify the password that is used to access the truststore file.
- Optionally, set the Enable FIPS (EnableFIPS) connection option to `1` to allow the driver to load the FIPS provider. The FIPS provider, introduced in OpenSSL 3.0, contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2. If you do not specify a value for Enable FIPS, the driver uses its default value (`0`) and loads the default provider.

Note:

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
- Do not set the Truststore Password connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.0-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms" for more information.

The following examples show how to configure the driver to establish a connection via user ID/password authentication and use data encryption via TLS/SSL server authentication. In these examples, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by the HostNameInCertificate option, and loads the FIPS provider for data encryption.

Connection string

Truststore:

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire Protocol;EnableFIPS=1;EncryptionMethod=1;
HostName=YourServer;HostNameInCertificate=MySubjectAltName;PortNumber=5439;
Database=db1;Truststore=TrustStoreName;ValidateServerCertificate=1
```

Note: On Windows, the driver validates the server certificate against the root certificates available in both truststore and Windows certificate store. If a matching certificate is found in either of the stores, the connection is established.

Windows certificate store:

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire Protocol;EnableFIPS=1;EncryptionMethod=1;
HostName=YourServer;HostNameInCertificate=MySubjectAltName;PortNumber=5439;
Database=db1;ValidateServerCertificate=1
```

Note: The LogonID 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.

odbc.ini

Truststore:

```
Driver=ODBCHOME/lib/ivrsftxx.so
Description=DataDirect Amazon Redshift Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=5439
...
Database=db1
...
Truststore=TrustStoreName
...
ValidateServerCertificate=1
...
```

Note: On Windows, the driver validates the server certificate against the root certificates available in both truststore and Windows certificate store. If a matching certificate is found in either of the stores, the connection is established.

Windows certificate store:

```
Driver=ODBCHOME/lib/ivrsftxx.so
Description=DataDirect Amazon Redshift Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=5439
...
Database=db1
...
ValidateServerCertificate=1
...
```

Note: The LogonID 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 also

Connection option descriptions on page 93

Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms on page 81

TLS/SSL client authentication

If the server is configured for TLS/SSL client authentication, the server asks the client to verify its identity after the server identity has been proven. Similar to server authentication, the client sends a public certificate to the server to accept or deny. The client stores its public certificate in an encrypted file known as a *keystore*. Public certificates are paired with a private key in the keystore. To send the public certificate, the driver must access the private key.

Like the truststore, most keystores are password-protected. The driver must be able to locate the keystore and unlock the keystore with the appropriate password. Two connection options are available to the driver to provide this information: Keystore (KeyStore) and Keystore Password (KeyStorePassword). The value of KeyStore is a pathname that specifies the location of the keystore file. The value of Keystore Password is the password required to access the keystore.

The private keys stored in a keystore can be individually password-protected. In many cases, the same password is used for access to both the keystore and to the individual keys in the keystore. It is possible, however, that the individual keys are protected by passwords different from the keystore password. The driver needs to know the password for an individual key to be able to retrieve it from the keystore. An additional connection option, Key Password (KeyPassword), allows you to specify a password for an individual key.

To configure the driver to use data encryption via TLS/SSL client authentication:

- Set the Host Name (HostName) option to specify the name or the IP address of the server to which you want to connect.
- Set the Port Number (PortNumber) option to specify the port number of the server listener. The default is `5439`.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Key Store (Keystore) option to specify the location of the keystore file.
- Set the Keystore Password (KeystorePassword) option to specify the password that is used to access the keystore file.
- Optionally, set the Enable FIPS (EnableFIPS) connection option to `1` to allow the driver to load the FIPS provider. The FIPS provider, introduced in OpenSSL 3.0, contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2. If you do not specify a value for Enable FIPS, the driver uses its default value (`0`) and loads the default provider.

  Note:

  - The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
  - For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.0-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms" for more information.
  - Do not set the Keystore Password connection option when using the FIPS provider. The keystore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.

The following examples show how to configure the driver to establish a connection via user ID/password authentication and use data encryption via TLS/SSL client authentication. In these examples, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate, and loads the FIPS provider for data encryption.

Connection string

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire Protocol;EnableFIPS=1;
EncryptionMethod=1;HostName=YourServer;
HostNameInCertificate=MySubjectAltName;PortNumber=5439;Database=Payroll;
Keystore=KeyStoreName;ValidateServerCertificate=1;
```

Note: The LogonID 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.

odbc.ini

```
Driver=ODBCHOME/lib/ivrsftxx.so
Description=DataDirect Amazon Redshift Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=5439
...
Database=Payroll;
...
KeyStore=KeyStoreName
...
ValidateServerCertificate=1
...
```

Note: The LogonID 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 also

Connection option descriptions on page 93

Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms on page 81

Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms

For using the OpenSSL 3.0 providers (FIPS and default), the certificates for TLS/SSL encryption must be generated using the OpenSSL 3.0-compliant cryptographic algorithms.

There are multiple ways of generating these certificates. The following commands demonstrate one of them. You can use these commands to generate the certificates and add them to the truststore and keystore files.

Note: The openssl.exe file is required for running these commands. You can download it from the official OpenSSL website.

Truststore:

```
openssl.exe pkcs12 -in certificate_name -export -out truststore_filename -nokeys
-keypbe cryptographic_algorithm -certpbe cryptographic_algorithm -password
pass:truststore_password -nomac
```

where:

`certificate_name`

is the name of the certificate you are generating.

`truststore_filename`

is the name of the truststore file.

`cryptographic_algorithm`

is the cryptographic algorithm you are using to generate the certificate.

`truststore_password`

is the password required for accessing the truststore file.

Example:

```
openssl.exe pkcs12 -in nc-thunder-SHA256.cer -export -out truststorepw.pfx -nokeys -keypbe
 AES-256-CBC -certpbe AES-256-CBC -password pass:MyPassW0rd -nomac
```

Keystore:

```
openssl.exe pkcs12 -in certificate_name -inkey privatekey_file -export -out
keystore_file -keypbe cryptographic_algorithm -certpbe cryptographic_algorithm
-nomac
```

where:

`certificate_name`

is the name of the certificate you are generating.

`privatekey_file`

is the name of the file that contains the private key.

`truststore_filename`

is the name of the keystore file.

`cryptographic_algorithm`

is the cryptographic algorithm you are using to generate the certificate.

Example:

```
openssl.exe pkcs12 -in nc-thunder-SHA256.cer -inkey ./file.pem -export -out keystorepw.pfx
 -keypbe AES-256-CBC -certpbe AES-256-CBC -nomac
```

Note: If you are using the Windows certificate store for TLS/SSL encryption, import the certificates generated with the OpenSSL 3.0-compliant algorithms into the store.

Designating an OpenSSL library

Important: Currently, the driver supports only version 3.0 of the OpenSSL library.

The driver uses OpenSSL library files (TLS/SSL Support Files) to implement cryptographic functions for data sources or connections when encrypting data. By default, the driver is configured to use the most secure version of the library installed with the product; however, you can designate a different version to address security vulnerabilities or incompatibility issues with your current library. Although the driver is only certified against libraries provided by Progress, you can also designate libraries that you supply. The methods described in this section can be used to designate an OpenSSL library file.

Note: For the default library setting, current information, and a complete list of installed OpenSSL libraries, refer to the readme file installed with your product.

File replacement

In the default configuration, the drivers use the OpenSSL library file located in the `\drivers` subdirectory for Windows installations and the `/lib` subdirectory for UNIX/Linux. You can replace this file with a different library to change the version used by the drivers. When using this method, the replacement file must contain both the cryptographic and TLS/SSL libraries and use the same file name as the default library. For example, the latest version of the library files use the following naming conventions:

Windows:

- Version 3.0: *xx*`openssl30.dll`

UNIX/Linux:

- Version 3.0: *xx*`openssl30.so` `[.sl]`

Designating the absolute path to a library

For libraries that do not use the default directory structure or file names, you must specify the absolute path to your cryptographic library for the CryptoLibName (CryptoLibName) option and the absolute path to your TLS/SSL library for the SSLLibName (SSLLibName) option. If you are using OpenSSL library files provided by Progress, these libraries are combined into a single file; therefore, the value specified for these options should be the same. For non-Progress library files, the libraries may use separate files, which would require specifying the unique paths to the `libeay32.dll` (cryptographic library) and `ssleay32.dll` (TLS/SSL library) files.

If you are using a GUI, these options are not exposed on the setup dialog. Instead, use the Extended Options field on the Advanced tab to configure these options. See "CryptoLibName" and "SSLLibName" for details.

See also

CryptoLibName on page 109
SSLLibName on page 135

Summary of security-related options

The following table summarizes how security-related connection options work with the driver. The connection options are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name is listed immediately after the GUI name in parentheses. See "Connection option descriptions" for details about configuring the options.

Table 3: Summary: Data Encryption Connection Options

Option	Description
Crypto Protocol Version (CryptoProtocolVersion)	Specifies the cryptographic protocols to use when SSL is enabled using the Encryption Method connection option (`EncryptionMethod=1` | `6`). **Default:** `TLSv1.2, TLSv1.1, TLSv1`
CryptoLibName (CryptoLibName)	The absolute path for the OpenSSL library file containing the cryptographic library to be used by the data source or connection when SSL is enabled. The cryptograpic library contains the implementations of cryptographic algorithms the driver uses for data encryption. **Default:** Empty string
Enable FIPS on page 111	Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled (`Encryption Method=1`). If disabled, the OpenSSL library uses cryptographic algorithms from the default provider. If enabled, the OpenSSL library uses cryptographic algorithms from the FIPS provider. **Default:** Disabled
Encryption Method (EncryptionMethod)	The method the driver uses to encrypt data sent between the driver and the database server. If set to `0` (No Encryption), data is not encrypted. If set to `1` (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option. If set to `6` (RequestSSL), the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. **Default:** `0` (No Encryption)
Host Name In Certificate (HostNameInCertificate)	A host name for certificate validation when SSL encryption is enabled (`Encryption Method=1` | `6`) and validation is enabled (`Validate Server Certificate=1`). **Default:** None
Key Password (KeyPassword)	Specifies the password used to access the individual keys in the keystore file when SSL is enabled (`Encryption Method=1` | `6`) and SSL client authentication is enabled on the database server. **Default:** None
Key Store (Keystore)	The name of the directory containing the keystore file to be used when SSL is enabled (`EncryptionMethod=1` | `6`) and SSL client authentication is enabled on the database server. **Default:** None

Option	Description
Key Store Password (KeystorePassword)	The password used to access the keystore file when SSL is enabled (`Encryption Method=1` | `6`) and SSL client authentication is enabled on the database server. **Default:** None
SSLLibName (SSLLibName)	The absolute path for the OpenSSL library file containing the SSL library to be used by the data source or connection when SSL is enabled. The SSL library contains the implementations of SSL protocols the driver uses for data encryption. **Default:** Empty string
Trust Store (Truststore)	The directory that contains the truststore file and the truststore file name to be used when SSL is enabled (`EncryptionMethod=1` | `6`) and server authentication is used. **Default:** None
Trust Store Password (TruststorePassword)	Specifies the password that is used to access the truststore file when SSL is enabled (`EncryptionMethod=1` | `6`) and server authentication is used. **Default:** None
User Name (LogonID)	The default user ID used to connect to your database.
Validate Server Certificate (ValidateServerCertificate)	If enabled, the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. If disabled, the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options. **Default:** Enabled

See also

Connection option descriptions on page 93

Authentication

The driver supports the following authentication methods:

- *Basic authentication* authenticates the user to the database using the specified user IDs and passwords.
- *Azure Active Directory authentication* authenticates the user to the database using an Active Directory user name and password.

By default, the driver is configured to use Basic authentication (`AuthenticationMethod=0`).

See also

Authentication Method on page 99

Basic authentication

Basic authentication is the default authentication method.

To configure the driver to use basic authentication:

- Set the Host Name (HostName) option to specify the IP address endpoint of the Amazon Redshift cluster to which you want to connect.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect.
- Set the User Name (LogonID) option to specify your user name.
- Set the Password option to specify your password.

Note: The User Name 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.

The following examples show the connection information required to establish a connection using Basic.

Connection string

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire Protocol;Database=Redshift2;
HostName=RedshiftServer;LogonID=John;Password=secret;PortNumber=5439;
```

odbc.ini

```
[Amazon Redshift]
Driver=ODBCHOME/lib/ivrsft28.so
...
Description=DataDirect 8.0 Amazon Redshift Wire Protocol
...
Database=Redshift2
...
HostName=RedshiftServer
...
LogonID=John
...
Password=secret
...
PortNumber=5439
...
```

Azure Active Directory authentication

The driver supports Azure Active Directory authentication (Azure AD). Azure AD authentication allows administrators to centrally manage user permissions to Amazon Redshift. When Azure AD authentication is enabled, all communications to Amazon Redshift are encrypted.

To configure the driver to use Azure AD authentication:

- Set the Authentication Method (AuthenticationMethod) option to `13` (AzureAD).
- Set the Azure Client ID (AzureClientID) option to specify the client ID key for your application.
- Set the Azure Client Secret (AzureClientSecret) option to specify the client secret for your application.

- Set the Azure Tenant ID (AzureTenantID) option to specify the ID of the Azure AD instance in which you create and manage your application.
- Set the AWS DB User (AWSDBUser) option to specify your Amazon Web Services (AWS) user name.

  Note: You can create a new AWS user using the Setup dialog box. To create a new AWS user, on the Authentication tab, specify the name for the new AWS user in the AWS DB User field, and then select the Auto create check box (`AutoCreate=1`).

- Optionally, set the AWS DB Group (AWSDBGroup) option to specify the name of the AWS user group your AWS user name is a part of.
- Specify values for one of the following sets of options:
  - AWS Region and AWS Cluster:
    - Set the AWS Region (AWSRegion) option to specify the name of the region that hosts your AWS server. For example, `us-east-1` or `us-east-2`.
    - Set the AWS Cluster (AWSCluster) option to specify the name of the Amazon Redshift cluster that contains the database you want to connect to.
  - Host Name and Port Number:
    - Set the Host Name (HostName) option to specify the IP address endpoint of the Amazon Redshift cluster to which you want to connect.
    - Set the Port Number (PortNumber) option to specify the port number of the server listener.

  Note: If values are specified for both the sets of options, the values for Host Name and Port Number take precedence over those for AWS Region and AWS Cluster.

- Set the Database option to specify the name of the database to which you want to connect.
- Set the User Name (LogonID) option to specify your user name.
- Set the Password option to specify your password.

Note: The User Name 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.

The following examples show the connection information required to establish a connection using the Azure AD authentication. These examples use the values for Port Number and Host Name instead of those for AWS Region and AWS Cluster.

Connection string

```
DRIVER=DataDirect 8.0 Amazon Redshift Wire Protocol;AuthenticationMethod=13;
AWSDBUser=jsmith;AWSDBGroup=awsgroup1;AzureClientID=abc123;
AzureClientSecret=abc456;AzureTenantID=xyz456;HostName=RedshiftServer;
LogonID=John;Password=secret;PortNumber=5432;
```

odbc.ini file

```
[Amazon Redshift]
Driver=ODBCHOME/lib/ivrsft28.so
...
```

```
Description=DataDirect 8.0 Amazon Redshift Wire Protocol
...
AuthenticationMethod=13
...
AWSDBUser=jsmith;
...
AWSDBGroup=awsgroup1;
...
AzureClientID=abc123;
...
AzureClientSecret=abc456;
...
AzureTenantID=xyz456;
...
HostName=RedshiftServer
...
LogonID=John
...
Password=secret
...
PortNumber=5432;
...
```

See also

Authentication tab on page 60

Connection option descriptions on page 93

Isolation and lock levels supported

Amazon Redshift supports isolation level 0 (read uncommitted), level 1 (read committed), 2 (Repeatable read), and level 3 (serializable). Amazon Redshift supports record-level locking.

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

Unicode support

The Amazon Redshift Wire Protocol driver automatically determines whether the Amazon Redshift database is a Unicode database.

Binding parameter markers

An ODBC application can prepare a query that contains dynamic parameters. Each parameter in a SQL statement must be associated, or bound, to a variable in the application before the statement is executed. When the application binds a variable to a parameter, it describes that variable and that parameter to the driver. Therefore, the application must supply the following information:

- The data type of the variable that the application maps to the dynamic parameter
- The SQL data type of the dynamic parameter (the data type that the database system assigned to the parameter marker)

The two data types are identified separately using the SQLBindParameter function. You can also use descriptor APIs as described in the Descriptor section of the ODBC specification (version 3.0 and higher).

The driver relies on the binding of parameters to know how to send information to the database system in its native format. If an application furnishes incorrect parameter binding information to the ODBC driver, the results will be unpredictable. For example, the statement might not be executed correctly.

To ensure interoperability, your driver uses only the parameter binding information that is provided by the application.

Persisting a Result Set as an XML Data File

Your driver allows you to persist a result set as an XML data file with embedded schema. To implement XML persistence, a client application must do the following:

1. Turn on STATIC cursors. For example:

   ```
   SQLSetStmtAttr (hstmt, SQL_ATTR_CURSOR_TYPE, SQL_CURSOR_STATIC, SQL_IS_INTEGER)
   ```

   Note: A result set can be persisted as an XML data file only if the result set is generated using STATIC cursors. Otherwise, the following error is returned:

   ```
   Driver only supports XML persistence when using driver’s static cursors.
   ```

2. Execute a SQL statement. For example:

   ```
   SQLExecDirect (hstmt, "SELECT * FROM GTABLE", SQL_NTS)
   ```

3. Persist the result set as an XML data file. For example:

   ```
   SQLSetStmtAttr (hstmt, SQL_PERSIST_AS_XML, "C:\temp\GTABLE.XML", SQL_NTS)
   ```

   Note: A statement attribute is available to support XML persistence, SQL_PERSIST_AS_XML. A client application must call SQLSetStmtAttr with this attribute as an argument. See the following table for the definition of valid arguments for SQLSetStmtAttr.

Argument	Definition
StatementHandle	The handle of the statement that contains the result set to persist as XML.
Attribute	SQL_PERSIST_AS_XML. This statement attribute can be found in the file qesqlext.h, which is installed with the driver.
ValuePtr	Pointer to a URL that specifies the full path name of the XML data file to be generated. The directory specified in the path name must exist, and if the specified file name exists, the file will be overwritten.
StringLength	The length of the string pointed to by ValuePtr or SQL_NTS if ValuePtr points to a NULL-terminated string.

A client application can choose to persist the data at any time that the statement is in an executed or cursor-positioned state. At any other time, the driver returns the following message:

```
Function Sequence Error
```

Packet logging

The driver code includes a packet logging mechanism that allows you to log TCP packets transmitted between your driver and database over the network layer. The logs compiled from can then be analyzed and used to troubleshoot issues. You can enable and configure logging using driver connection options.

Note: The packet logging mechanism is supported only for drivers that transmit TCP packets. Refer to "Packet Logging" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported drivers.

See the following "Packet Logging Connection options" section for a list of connection options used to configure packet logging.

To enable TCP packet logging:

1. Configure and enable packet logging using one of the following methods:
   - Driver setup dialog (Windows)
   - odbc.ini file (UNIX/Linux)
   - Connection string

   See the following "Configuring and enabling packet logging" section for details.
2. Start your application and reproduce the issue.
3. Stop the application and disable packet logging.
4. Send your logs to Technical Support for analysis. Optionally, you can view your logs using a text editor.

Configuring and enabling packet logging

The following driver configuration methods can be used to enable and configure packet logging. Note that only the `EnablePacketLogging` connection option is required to enable packet logging. If you do not specify values for the other connection options for packet logging, the default behavior is used.

Driver setup dialog (Windows)

You can specify connection options for packet logging in the Extended Options field of the **Advanced** tab. For example:

```
EnablePacketLogging=1;PacketLoggingFilePrefix=C:\temp\myPacketLog;
PacketLoggingMaxFileSize=7500
```

odbc.ini file (UNIX/Linux)

In your data source definition in the `[ODBC Data Sources]` section of the system information file, you can specify connection options that control packet logging.

```
[Amazon Redshift Wire Protocol]
Driver=ODBCHOME/lib/ivrsft28.so
Description=DataDirect 8.0 Amazon Redshift Wire Protocol
...
Database=MyDB
...
EnablePacketLogging=1
...
HostName=RedshiftServer
...
LogonID=JOHN
...
PacketLoggingFilePrefix=/tmp/myPacketLog
...
PacketLoggingMaxFileSize=102400
...
PacketLoggingMaxNumFiles=10
...
Password=secret
...
PortNumber=5439
...
```

Connection string

You can specify connection options that configure packet logging in connection strings.

```
DRIVER={DataDirect 8.0 Amazon Redshift Wire
Protocol};HostName=RedshiftServer;PortNumber=5439;
LogonID=JOHN;Password=secret;Database=MyDB;EnablePacketLogging=1;
PacketLoggingFilePrefix=C:\temp\myPacketLog;
```

Packet logging connection options

The following table describes the connection options used to configure packet logging.

Table 4: Packet Logging Connection Options

Option	Description
`EnablePacketLogging`	If set to `0`, packet logging is disabled. This is the default. If set to `1`, packet logging is enabled. If set to `2`, packet logging is enabled, but the generated log file does not contain packet data. This value is typically used for performance testing. (Windows only) If set to `5`, packet logging and ODBC tracing are enabled. If set to `6`, packet logging and ODBC tracing are enabled, but the log file for packet logging does not contain data.

Option	Description
`PacketLoggingFlush`	If set to `0`, the operating system determines when to write the log content stored in memory to disk. This is the default. If set `1`, the driver determines when to write the log content stored in memory to disk. If set to `2`, the content of memory is written to a the log file after each write. This setting provides a more complete logging history in the event of a crash, but can incur a performance penalty.
`PacketLoggingFilePrefix`	Specifies the path and prefix name of the log file. If no path is specified, the trace log resides in the working directory of the application you are using. For example: • `/tmp/myLogFile` (UNIX/Linux) • `C:\temp\myLogFile` (Windows) The above examples would generate a file named `myLogFile`*`YYYYMMDDhhmmssxxx_nn`*`.out` in the `temp` directory. If you do not specify a value for this option, the driver creates log files in the working directory using the following form: `pkt`*`YYYYMMDDhhmmssxxx_nn`*`.out`.
`PacketLoggingMaxFileSize`	Specifies the file size limit (in KB) of the log file. Once this file size limit is reached, a new log file is created and logging continues. The default is `102400`. Note that subsequent files are named by appending sequential numbers, starting at `1`, to the end of the original file name, for example, `myLog<`*`timestamp`*`>_1.out`, `myLog<`*`timestamp`*`>_2.out`, and so on.
`PacketLoggingMaxNumFiles`	Specifies the maximum number of log files that can be created. The default is `10`. Once the maximum number of log files is created, the logging mechanism reopens the first file in the sequence, deletes the content, and continues logging in that file until the file size limit is reached, after which it repeats the process with the next file in the sequence.
`PacketLoggingMemBuffSize`	Specifies the maximum amount of memory, in kiloybtes, to use when writing packet logging. The default is `1024`.

5

Connection option descriptions

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

For example:

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

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, for example, Password, 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 table lists the connection string attributes supported by the Amazon Redshift Wire Protocol driver.

Table 5: Amazon Redshift Wire Protocol Driver Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASRV)	None

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
AuthenticationMethod (AM)	0 (Basic)
AutoCreate (ATCRT)	0
AWSCluster (ACLS)	None
AWSDBGroup (ADG)	None
AWSDBUser (ADU)	None
AWSRegion (ARG)	None
AzureClientID (AZCLID)	None
AzureClientSecret (AZCLISEC)	None
AzureTenantID (AZTNTID)	None
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CryptoLibName (CLN)	Empty string
CryptoProtocolVersion (CPV)	`TLSv1.2,TLSv1.1,TLSv1`
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
EnableDescribeParam (EDP)	1 (Enabled)
EnableFIPS (EF)	0 (Default provider)
EncryptionMethod (EM)	0 (No Encryption)
ExtendedColumnMetaData (ECMD)	0 (Disabled)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)

Attribute (Short Name)	Default
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) (UNIX ONLY)	`4` (ISO 8559-1 Latin-1)
InitializationString (IS)	None
KeepAlive (KA)	`0` (Disabled)
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoadBalanceTimeout (LBT)	`0`
LoadBalancing (LB)	`0` (Disabled)
LoginTimeout (LT)	`15`
LogonID (UID)	None
Max Char Size	None
MaxPoolSize (MXPS)	`100`
MaxVarcharSize (MVS)	None
MinPoolSize (MNPS)	`0`
OpenSSLConfigFile (OSSLCNF)	*install_dir*`\drivers\openssl.cnf` (Windows) *install_dir*`/lib/openssl.cnf` (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*install_dir*`\drivers` (Windows) *install_dir*`/lib` (UNIX/Linux)
Password (PWD)	None
Pooling (POOL)	`0` (Disabled)
PortNumber (PORT)	`5439`
ProxyHost (PXHN)	Empty String
ProxyMode (PXM)	`0`

Attribute (Short Name)	Default
ProxyPassword (PXPW)	Empty String
ProxyPort (PXPT)	`0`
ProxyUser (PXU)	Empty String
QueryTimeout (QT)	`0`
ReportCodepageConversionErrors (RCCE)	`0` (Ignore Errors)
ShowSelectableTables (SST)	Enabled
SSLLibName (SLN)	Empty string
TransactionErrorBehavior (TEB)	`1` (Rollback)
Truststore (TS)	None
TruststorePassword (TSP)	None
ValidateServerCertificate (VSC)	`1` (Enabled)

For details, see the following topics:

- Alternate Servers
- Application Using Threads
- Authentication Method
- Auto Create
- AWS Cluster
- AWS DB Group
- AWS DB User
- AWS Region
- Azure Client ID
- Azure Client Secret
- Azure Tenant ID
- Connection Pooling
- Connection Reset
- Connection Retry Count
- Connection Retry Delay
- Crypto Protocol Version

- CryptoLibName
- Database Name
- Data Source Name
- Description
- Enable FIPS
- Enable SQLDescribeParam
- Encryption Method
- Extended Column MetaData
- Failover Granularity
- Failover Mode
- Failover Preconnect
- Fetch TSWTZ as Timestamp
- Host Name
- Host Name In Certificate
- IANAAppCodePage
- Initialization String
- Key Password
- Key Store
- Key Store Password
- Load Balance Timeout
- Load Balancing
- Login Timeout
- Max Char Size
- Max Pool Size
- Max Varchar Size
- Min Pool Size
- OpenSSL Config File
- OpenSSL Provider Path
- Password
- Port Number
- Proxy Host
- Proxy Mode
- Proxy Password

- Proxy Port
- Proxy User
- Query Timeout
- Report Codepage Conversion Errors
- Show Selectable Tables
- SSLLibName
- TCP Keep Alive
- Transaction Error Behavior
- Truststore
- Truststore Password
- User Name
- Validate Server Certificate

Alternate Servers

Attribute

AlternateServers (ASRV)

Purpose

A list of alternate database servers to which the driver tries to connect if the primary database server is unavailable. Specifying a value for this option enables connection failover for the driver. The value you specify must be in the form of a string that defines the physical location of each alternate server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection.

Valid Values

(`HostName=`*`hostvalue`*:`PortNumber=`*`portvalue`*:`Database=`*`databasevalue`*[, . . .])

You must specify the host name, port number, and database name of each alternate server.

Example

The following Alternate Servers value defines two alternate database servers for connection failover:

```
AlternateServers=(HostName=RedshiftServer:
PortNumber=5439:Database=Redshift1,
HostName=255.201.11.24:PortNumber=5440:Database=Pgredb2)
```

Notes

- An alternate server address in IPv6 format must be enclosed in double quotation marks.

Default

None

GUI Tab

Failover tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Default

`1` (Enabled)

GUI Tab

Advanced tab

See also

Performance considerations on page 64

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

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

Valid Values

`0` | `13`

Behavior

If set to `0` (Basic), the driver uses a hashed value, based on the concatenation of the user name and password, for authentication.

If set to `13` (AzureAD), the driver uses Azure AD authentication when establishing a connection.

Default Value

`0` (Basic)

GUI Tab

Auhthentication tab

See also

Authentication on page 85

Auto Create

Attribute

AutoCreate (ATCRT)

Purpose

Determines whether the driver creates a new AWS user while connecting to an Amazon Redshift database using Azure AD authentication (`AuthenticationMethod=13`).

Important: When creating a new AWS user, you must specify the name for the new AWS user in the AWS DB User field.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver creates a new AWS user. The value specified for the AWS DB User option is used as the name for the new AWS user.

If set to `0` (Disabled), the driver does not create a new AWS user.

Default

`0` (Disabled)

GUI Tab

Auhthentication tab

AWS Cluster

Attribute

AWSCluster (ACLS)

Purpose

Specifies the name of the Amazon Redshift cluster that contains the database you want to connect to. The driver uses this value when connecting to an Amazon Redshift database using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

`string`

is the name of the Amazon Redshift cluster that contains the database you want to connect to.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

AWS DB Group

Attribute

AWSDBGroup (ADG)

Purpose

Specifies the name of AWS user group that your AWS user name is a part of. The driver uses this value when authenticating to Amazon Redshift using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

string

is the name of the AWS user group that your AWS user name is a part of.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

AWS DB User

Attribute

AWSDBUser (ADU)

Purpose

Specifies your AWS user name when authenticating to Amazon Redshift using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

string

where:

string

is your AWS user name.

Notes

- You can create a new AWS user using the Setup dialog box. To create a new AWS user, on the Authentication tab, specify the name for the new AWS user in the AWS DB User field, and then select the Auto create check box (`AutoCreate=1`).

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

AWS Region

Attribute

AWSRegion (ARG)

Purpose

Specifies the name of the region that hosts your AWS server. The driver uses this value when connecting to an Amazon Redshift database using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

`string`

is the name of the region that hosts your AWS server. For example,`us-east-1` or `us-east-2`.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

Azure Client ID

Attribute

AzureClientID (AZCLID)

Purpose

Specifies the client ID key for your application when authenticating to Amazon Redshift using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

`string`

is the client ID key for your application.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

Azure Client Secret

Attribute

AzureClientSecret (AZCLISEC)

Purpose

Specifies the client secret for your application when authenticating to Amazon Redshift using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

`string`

is the client secret for your application.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

Azure Tenant ID

Attribute

AzureTenantID (AZTNTID)

Purpose

Specifies the ID of the Azure AD instance in which you create and manage your application. The driver uses this value when authenticating to Amazon Redshift using Azure AD authentication (`AuthenticationMethod=13`).

Valid Values

`string`

where:

`string`

is the Tenant ID for your application.

Default Value

No default value

GUI Tab

Auhthentication tab

See also

Azure Active Directory authentication on page 86

Connection Pooling

Attribute

Pooling (POOL)

Purpose

Specifies whether to use the driver's connection pooling.

Valid Values

`0` | `1`

Behavior

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

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

Notes

- The application must be thread-enabled to use connection pooling.
- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Pooling tab

See also

Performance considerations on page 64

Connection Reset

Attribute

ConnectionReset (CR)

Purpose

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

Valid Values

`0 | 1`

Behavior

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

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

Notes

- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Pooling tab

See also

Performance considerations on page 64

Connection Retry Count

Attribute

ConnectionRetryCount (CRC)

Purpose

The number of times the driver retries connection attempts to the primary database server, and if specified, alternate servers until a successful connection is established.

This option and the Connection Retry Delay connection option, which specifies the wait interval between attempts, can be used in conjunction with failover.

Valid Values

`0` | *x*

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

`0`

GUI Tab

Failover tab

Connection Retry Delay

Attribute

ConnectionRetryDelay (CRD)

Purpose

Specifies the number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer.

This option and the Connection Retry Count connection option can be used in conjunction with failover.

Valid Values

`0` | *x*

where

x

is a positive integer from 1 to 65535.

Behavior

If set to `0`, there is no delay between retries.

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

Default

`3`

GUI Tab

Failover tab

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

Specifies a comma-separated list of the cryptographic protocols to use when SSL is enabled using the Encryption Method connection option (`EncryptionMethod=1` | `6`). When multiple protocols are specified, the driver uses the highest version supported by the server. If none of the specified protocols are supported by the database server, the connection fails and the driver returns an error.

Valid Values

`cryptographic_protocol` `[[,` *`cryptographic_protocol`* `]...]`

where:

`cryptographic_protocol`

is one of the following cryptographic protocols:

`TLSv1.2` | `TLSv1.1` | `TLSv1`

Example

If your security environment is configured to use TLSv1.2 and TLSv1.1, specify the following values:

```
CryptoProtocolVersion=TLSv1.2, TLSv1.1
```

Default

`TLSv1.2,TLSv1.1,TLSv1`

GUI Tab

Security tab

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

The absolute path for the OpenSSL library file containing the cryptographic library to be used by the data source or connection when TLS/SSL is enabled. The cryptograpic library contains the implementations of cryptographic algorithms the driver uses for data encryption.

This option allows you to designate a different cryptographic library if you encounter issues with the default version or want to use a library that you provide. Common issues that require designating a different library include security vulnerabilities with specific libraries or compatibility issues with your server or application.

Valid Values

`absolute_path\openssl_filename`

where:

`absolute_path`

is the absolute path to where the OpenSSL file is located

`openssl_filename`

is the name of the OpenSSL library file containing the cryptographic library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll
```

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the CryptoLibName option. The CryptoLibName option provides a method for you to specify a cryptographic library file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.

Default

Empty string

GUI Tab

The value for this option is specified as an option-value pair in the Extended Options field on the Advanced tab. For example:

```
CryptoLibName=C:\Program Files\Progress\DataDirect\ODBC\drivers\ddopenssl30.dll;
```

See also

- SSLLibName on page 135
- Advanced tab on page 48

Database Name

Attribute

Database (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.

Default

None

GUI Tab

General tab

Data Source Name

Attribute

DataSourceName (DSN)

Description

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

Valid Values

`string`

where:

`string`

is the name of a data source.

Default

None

GUI Tab

General tab

Description

Attribute

Description (n/a)

Purpose

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

Valid Values

`string`

where

`string`

is a description of a data source.

Default

None

GUI Tab

General tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled.

Valid Values

`0 | 1`

Behavior

If set to `0`, the OpenSSL library uses cryptographic algorithms from the default provider.

If set to `1`, the OpenSSL library uses cryptographic algorithms from the FIPS provider.

Notes

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit. On the other platforms, the driver uses the default provider of the OpenSSL 3.0 library.
- Do not set the Truststore Password (TruststorePassword) connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be encrypted with the OpenSSL 3.0-compliant cryptographic algorithms. See "Generating TLS/SSL certificates with OpenSSL 3.0-compliant algorithms" for more information.

Default

`0`

See also

TLS/SSL server authentication on page 76
TLS/SSL client authentication on page 79
Generating TLS/SSL certificates using OpenSSL 3.0-compliant algorithms on page 81

Enable SQLDescribeParam

Attribute

EnableDescribeParam (EDP)

Purpose

Determines whether SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

Valid Values

`0 | 1`

Behavior

If set to `1` (enabled), SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

If set to `0` (disabled), the driver does not support SQLDescribeParam and returns the message: `Driver does not support this function.`

Default

`1` (Enabled)

GUI Tab

Advanced tab

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

The method the driver uses to encrypt data sent between the driver and the database server. It supports the RequestSSL functionality. When RequestSSL is enabled, login requests and data are encrypted if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established.

Valid Values

`0| 1| 6`

Behavior

If set to `0` (No Encryption), data is not encrypted.

If set to `1` (SSL), If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.. If the server is not configured for SSL, the connection fails.

If set to `6` (RequestSSL), the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. The SSL protocol used is determined by the setting of the Crypto Protocol Version connection option.

Notes

- This connection option can affect performance.

Default

`0` (No Encryption)

GUI Tab

Security tab

See also

Performance considerations on page 64

Extended Column MetaData

Attribute

ExtendedColumnMetaData (ECMD)

Purpose

Determines how the driver returns column metadata when using SQLDescribeCol and SQLColAttribute.

Valid Values

`0 | 1`

Behavior

If set to `1` (Enabled), SQLDescribeCol returns the actual values for Data Type, Column Size, Decimal Digits, and Nullable. SQLColAttribute returns the actual values for:

- SQL_DESC_CATALOG_NAME: *`catalog_name`*
- SQL_DESC_TABLE_NAME: *`table_name`*
- SQL_DESC_BASE_COLUMN_NAME: *`base_column_name`*
- SQL_DESC_LOCAL_TYPE_NAME: *`local_type_name`*
- SQL_DESC_NULLABLE: *`nullable`*
- SQL_DESC_AUTO_UNIQUE_VALUE: *`auto_unique_value`*

If set to `0` (Disabled), SQLDescribeCol returns the Data Type, Column Size, and Decimal Digits for the column. The value SQL_NULLABLE_UNKNOWN is returned for Nullable. SQLColAttribute returns the following attribute values:

- SQL_DESC_CATALOG_NAME: empty string
- SQL_DESC_TABLE_NAME: empty string
- SQL_DESC_BASE_COLUMN_NAME: empty string
- SQL_DESC_LOCAL_TYPE_NAME: empty string
- SQL_DESC_NULLABLE: SQL_NULLABLE_UNKNOWN
- SQL_DESC_AUTO_UNIQUE_VALUE: SQL_FALSE

Default

`0` (Disabled)

GUI Tab

Advanced tab

Failover Granularity

Attribute

FailoverGranularity (FG)

Purpose

Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection.

This option applies only when Failover Mode is set to 1 (Extended Connection) or 2 (Select).

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

`0 | 1 | 2 | 3`

Behavior

If set to `0` (Non-Atomic), the driver continues with the failover process and posts any errors on the statement on which they occur.

If set to `1` (Atomic) the driver fails the entire failover process if an error is generated as the result of anything other than executing and repositioning a Select statement. If an error is generated as a result of repositioning a result set to the last row position, the driver continues with the failover process, but generates a warning that the Select statement must be reissued.

If set to `2` (Atomic Including Repositioning), the driver fails the entire failover process if any error is generated as the result of restoring the state of the connection or the state of work in progress.

If set to `3` (Disable Integrity Check), the driver does not verify that the rows that were restored during the failover process match the original rows. This value applies only when Failover Mode is set to 2 (Select).

Default

`0` (Non-Atomic)

GUI Tab

Failover tab

Failover Mode

Attribute

FailoverMode (FM)

Purpose

Specifies the type of failover method the driver uses.

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

`0 | 1 | 2`

Behavior

If set to `0` (Connection), the driver provides failover protection for new connections only.

If set to `1` (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress.

If set to `2` (Select), the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed.

Default

`0` (Connection)

GUI Tab

Failover tab

Failover Preconnect

Attribute

FailoverPreconnect (FP)

Purpose

Specifies whether the driver tries to connect to the primary and an alternate server at the same time.

This attribute applies only when Failover Mode is set to `1` (Extended Connection) or `2` (Select) and at least one alternate server is specified.

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

`0 | 1`

Behavior

If set to `0` (Disabled), the driver tries to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection. This value provides the best performance, but your application typically experiences a short wait while the failover connection is attempted.

If set to `1` (Enabled), the driver tries to connect to the primary and an alternate server at the same time. This can be useful if your application is time-sensitive and cannot absorb the wait for the failover connection to succeed.

Default

`0` (Disabled)

GUI Tab

Failover tab

Fetch TSWTZ as Timestamp

Attribute

FetchTSWTZasTimestamp (FTSWTZAT)

Purpose

Determines whether the driver returns column values with the TimestampTZ data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver returns column values with the TimestampTZ data type as the ODBC type SQL_TYPE_TIMESTAMP. The time zone information in the fetched value is truncated. Use this value if your application needs to process values the same way as TIMESTAMP columns.

If set to `0` (Disabled), the driver returns column values with the TimestampTZ data type as the ODBC data type SQL_VARCHAR. Use this value if your application requires the time zone information in the fetched value.

Default

`0` (Disabled)

GUI Tab

Advanced tab

Host Name

Attribute

HostName (HOST)

Purpose

The endpoint for the Amazon Redshift cluster to which you want to connect.

Valid Values

`IP_address`

where:

`IP_address`

is the IP address endpoint of the cluster to which you want to connect.

The IP address can be specified in either IPv4 or IPv6 format.

Example

`199.226.224.34`.

Default

None

GUI Tab

General tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when SSL encryption is enabled (`Encryption Method=1`) and validation is enabled (`Validate Server Certificate=1`). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values

`host_name` `|` *`#SERVERNAME#`*

where:

`host_name`

is the host name specified in the certificate. Consult your SSL administrator for the correct value.

Behavior

If set to a host name, the driver compares the specified host name to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name with the Common Name (CN) part of the certificate. If the values do not match, the connection fails and the driver throws an exception.

If set to *#SERVERNAME#*, the driver compares the server name that is specified in the connection URL or data source of the connection to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name to the CN part of the certificate's Subject name. If the values do not match, the connection fails and the driver throws an exception. If multiple CN parts are present, the driver validates the host name against each CN part. If any one validation succeeds, a connection is established.

Default

None

GUI Tab

Security tab

IANAAppCodePage

UNIX®

Attribute

IANAAppCodePage (IACP)

Purpose

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

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

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

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

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

Valid Values

`IANA_code_page`

where:

`IANA_code_page`

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

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

N/A

See Also

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

Initialization String

Attribute

InitializationString (IS)

Purpose

A SQL command that is issued immediately after connecting to the database to manage session settings.

Valid Values

`SQL_command`

where:

`SQL_command`

is a valid SQL command that is supported by the database. Separate multiple commands by semicolons. If this option is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified.

Example

To set the date format on every connection, specify:

```
InitializationString=Set DateStyle='ISO, MDY'
```

Notes

- If the statement fails to execute, the connection fails and the driver reports the error returned from the server.

Default

None

GUI Tab

Advanced tab

Key Password

Attribute

KeyPassword (KP)

Purpose

The password used to access the individual keys in the keystore file when SSL is enabled (`Encryption Method=1`) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values

`key_password`

where:

`key_password`

is the password of a key in the keystore.

Default

None

GUI Tab

Security tab

Key Store

Attribute

Keystore (KS)

Purpose

The name of the directory containing the keystore file to be used when SSL is enabled (`Encryption Method=1`) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request. If you do not specify a directory, the current directory is used.

Valid Values

`keystore_directory`

where:

`keystore_directory`

is the location of the keystore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Keystore option. The Keystore option provides a method for you to specify a keystore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The keystore and truststore files can be the same file.

Default

None

GUI Tab

Security tab

Key Store Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when SSL is enabled (`Encryption Method=1`) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

Valid Values

`keystore_password`

where:

`keystore_password`

is the password of the keystore file.

Notes

- The keystore and truststore files may be the same file; therefore, they may have the same password.

Default

None

GUI Tab

Security tab

Load Balance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

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

Valid Values

`0` | *`x`*

where:

`x`

is a positive integer that specifies a number of seconds.

Behavior

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

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

Notes

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

Default

`0`

GUI Tab

Pooling tab

See also

Performance considerations on page 64

Load Balancing

Attribute

LoadBalancing (LB)

Purpose

Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values

`0 | 1`

Behavior

If set to `1` (Enabled), the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order.

If set to `0` (Disabled), the driver does not use client load balancing and connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified).

Notes

- This option has no effect unless alternate servers are defined for the Alternate Servers connection option.

Default

`0` (Disabled)

GUI Tab

Failover tab

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 represents 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

`15`

GUI Tab

Advanced tab

Max Char Size

Attribute

MaxCharSize (MCS)

Purpose

Specifies the maximum size of columns of type SQL_CHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from `1` to `4096`

Behavior

When not specified, the actual size of the columns from the database is persisted to the application.

If you specify a value that is not in the specified range, the driver uses the maximum value of the SQL_CHAR data type, 4096.

Default

None. The actual size of the columns from the database is persisted to the application.

GUI Tab

Advanced tab

Max Pool Size

Attribute

MaxPoolSize (MXPS)

Purpose

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

Valid Values

An integer from `1` to `65535`

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

Notes

- This connection option can affect performance.

Default

`100`

GUI Tab

Pooling tab

See also

Performance considerations on page 64

Max Varchar Size

Attribute

MaxVarcharSize (MVS)

Purpose

Specifies the maximum size of columns of type SQL_VARCHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from `1` to *x*

where:

x

is maximum size of the SQL_VARCHAR data type.

Default

None. The actual size of the columns from the database is persisted to the application.

GUI Tab

Advanced tab

Min Pool Size

Attribute

MinPoolSize (MNPS)

Purpose

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

Valid Values

`0` | *`x`*

Behavior

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

Notes

- This connection option can affect performance.

Example

If set to `5`, the start-up number of connections in the pool is 5 in addition to the current existing connection.

Default

`0`

GUI Tab

Pooling tab

See also

Performance considerations on page 64

OpenSSL Config File

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

Specifies the absolute path to the configuration file required to load the FIPS provider when the driver is configured to use OpenSSL 3.0 with FIPS provider for TLS/SSL encryption (`EnableFIPS=1`).

Valid Values

`fips_config_file`

where:

`fips_config_file`

is the absolute path to the configuration file. For example: `/opt/Progress/DataDirect/ODBC/lib/openssl.cnf`.

Notes

- The OpenSSL Config File option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *install_dir*`\drivers\openssl.cnf` (Windows)
- *install_dir*`/lib/openssl.cnf` (UNIX/Linux)

OpenSSL Provider Path

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

Specifies the path to the directory that contains the provider library when TLS/SSL encryption is enabled.

Valid Values

provider_path

where:

provider_path

is the path to the directory that contains the provider library.

Notes

- The OpenSSL Provider Path option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *install_dir*`\drivers` (Windows)
- *install_dir*`/lib` (UNIX/Linux)

Password

Attribute

Password (PWD)

Purpose

The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

The port number of the server listener.

Valid Values

`port_name`

where:

`port_name`

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

Default

`5439`

GUI Tab

General tab

Proxy Host

Attribute

ProxyHost (PXHN)

Purpose

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

Valid Values

`server_name` | `IP_address`

where:

`server_name`

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

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

Default

Empty string

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy Host option is ignored.

GUI Tab

General tab

See Also

- Proxy Mode on page 130
- Proxy Password on page 131
- Proxy Port on page 132
- Proxy User on page 133

Proxy Mode

Attribute

ProxyMode (PXM)

Purpose

Determines whether the driver connects to an Amazon Redshift endpoint through an HTTP proxy server.

Valid Values

`0` | `1`

Behavior

If set to `0` (NONE), the driver connects directly to the Amazon Redshift endpoint specified by the Host Name connection option.

If set to `1` (HTTP), the driver connects to the Amazon Redshift endpoint through the HTTP proxy server specified by the ProxyHost connection option.

Default

`0` (NONE)

GUI Tab

General tab

See Also

- Proxy Host on page 129
- Host Name on page 117
- Proxy Password on page 131
- Proxy Port on page 132
- Proxy User on page 133

Proxy Password

Attribute

ProxyPassword (PXPW)

Purpose

Specifies the password needed to connect to the Proxy Server.

Valid Values

`String`

where:

`String`

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

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy Password option is ignored.
- Proxy Password is required only when the proxy server has been configured to require authentication.

Default

Empty string

GUI Tab

General tab

See Also

- Proxy Host on page 129
- Proxy Mode on page 130
- Proxy Port on page 132
- Proxy User on page 133

Proxy Port

Attribute

ProxyPort (PXPT)

Purpose

Specifies the port number where the Proxy Server is listening for HTTP requests.

Valid Values

`port_name`

where:

`port_name`

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

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy Port option is ignored.

Default

`0`

GUI Tab

General tab

See Also

- Proxy Host on page 129
- Proxy Mode on page 130
- Proxy Password on page 131
- Proxy User on page 133

Proxy User

Attribute

ProxyUser (PXU)

Purpose

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

Valid Values

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

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy User option is ignored.
- Proxy User is required only when the proxy server has been configured to require authentication.

Default

`Empty string`

GUI Tab

General tab

See Also

- Proxy Host on page 129
- Proxy Mode on page 130
- Proxy Password on page 131
- Proxy Port on page 132

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

The number of seconds for the default query timeout for all statements that are created by a connection. To override the value set by this connection option for an individual statement, set a different value in the SQL_ATTR_QUERY_TIMEOUT statement attribute on the SQLSetStmtAttr() function.

Valid Values

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

where:

x

is a positive integer that specifies 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

`0`

GUI Tab

Advanced tab

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

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

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

Valid Values

`0` | `1` | `2`

Behavior

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

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

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

Default

`0` (Ignore Errors)

GUI Tab

Advanced tab

Show Selectable Tables

Attribute

ShowSelectableTables (SST)

Purpose

Determines whether the driver returns metadata for all tables or only those for which the user has Select privileges. It can help the users with restricted Select privileges retrieve metadata for the required tables.

Valid Values

`0 | 1`

Behavior

If set to `0`, the driver returns metadata for all tables, irrespective of Select privileges.

If set to `1`, the driver returns metadata only for the tables for which the user has Select privileges.

Default

`1`

GUI Tab

Advanced tab

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

The absolute path for the OpenSSL library file containing the TLS/SSL library to be used by the data source or connection when TLS/SSL is enabled. The TLS/SSL library contains the implementations of TLS/SSL protocols the driver uses for data encryption.

This option allows you to designate a different TLS/SSL library if you encounter issues with the default version or want to use a library that you provide. Common issues that require designating a different library include security vulnerabilities with specific libraries or compatibility issues with your server or application.

Valid Values

`absolute_path\openssl_filename`

where:

`absolute_path`

is the absolute path to where the OpenSSL file is located

`openssl_filename`

is the name of the OpenSSL library file containing the TLS/SSL Library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll
```

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the SSLLibName option. The SSLLibName option provides a method for you to specify an OpenSSL library file used for SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.

Default

No default value

GUI Tab

The value for this option is specified as an option-value pair in the Extended Options field on the Advanced tab. For example:

```
SSLLibName=C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll;
```

See also

- Advanced tab on page 48
- CryptoLibName on page 109

TCP Keep Alive

Attribute

KeepAlive (KA)

Purpose

Specifies whether the driver enables TCPKeepAlive. TCPKeepAlive maintains idle TCP connections by periodically passing packets between the client and server. If either the client or server does not respond to a packet, the connection is considered inactive and is terminated. In addition, TCPKeepAlive prevents valid idle connections from being disconnected by firewalls and proxies by maintaining network activity.

Valid Values

`0 | 1`

Behavior

If set to `0` (Disabled), the driver does not enable TCPKeepAlive.

If set to `1` (Enabled), the driver enables TCPKeepAlive.

Default

`0` (Disabled)

GUI Tab

Advanced tab

Transaction Error Behavior

Attribute

TransactionErrorBehavior (TEB)

Purpose

Determines how the driver handles errors that occur within a transaction. When an error occurs in a transaction, the Redshift server does not allow any operations on the connection except for rolling back the transaction.

Valid Values

`0 | 1`

Behavior

If set to `0` (Do Nothing), the driver does not roll back the transaction when an error occurs. The application must handle the error and roll back the transaction. Any operation on the statement other than a rollback results in an error.

If set to `1` (Rollback), the driver rolls back the transaction when an error occurs. In addition to the original error message, the driver posts an error message indicating that the transaction has been rolled back.

Default

`1` (Rollback)

GUI Tab

Advanced tab

Truststore

Attribute

Truststore (TS)

Purpose

The directory that contains the truststore file and the truststore file name to be used when SSL is enabled (`Encryption Method=1`) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication. If you do not specify a directory, the current directory is used.

Valid Values

`truststore_directory\filename`

where:

`truststore_directory`

is the directory where the truststore file is located

`filename`

is the file name of the truststore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Truststore option. The Truststore option provides a method for you to specify a truststore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The keystore and truststore files may be the same file.

Default

None

GUI Tab

Security tab

Truststore Password

Attribute

TruststorePassword (TSP)

Purpose

The password that is used to access the truststore file when SSL is enabled (`Encryption Method=1`) and server authentication is used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

Valid Values

`truststore_password`

where:

`truststore_password`

is a valid password for the truststore file.

Notes

- The truststore and keystore files may be the same file; therefore, they may have the same password.

Default

None

GUI Tab

Security tab

User Name

Attribute

LogonID (UID)

Purpose

The default user ID that is used to connect to your database. Your ODBC application may override this value or you may override it in the logon dialog box or connection string.

Valid Values

`userid`

where:

`userid`

is a valid user ID with permissions to access the database.

Default

None

GUI Tab

Security tab

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (`Encryption Method=1`). When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate returned from the server even if the issuer is not a trusted CA is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

Truststore information is specified using the Trust Store and Trust Store Password options.

Valid Values

`0 | 1`

Behavior

If set to `1` (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

If set to `0` (Disabled), the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options.

Default

`1` (Enabled)

GUI Tab

Security tab on page 52