Progress DataDirect for ODBC for PostgreSQL Wire Protocol Driver

User's Guide

Release 8.0.2

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for PostgreSQL Wire Protocol Driver

The Progress® DataDirect® for ODBC™ for PostgreSQL™ Wire Protocol driver (the PostgreSQL Wire Protocol driver) supports PostgreSQL database servers.

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

For details, see the following topics:

- What's new in this release?
- Driver requirements
- Installing and setting up the driver (Windows)
- Installing and setting up the driver (UNIX/Linux)
- Connection string examples
- Version string information
- Data Types
- Driver specifications
- Additional information
- Troubleshooting

- Contacting Technical Support

What's new in this release?

For the latest certifications and enhancements, refer to the following:

- Release Notes
- Supported Configurations
- DatatDirect Support Matrices

Changes Since 8.0.2 GA

- **Driver Enhancements**
  - The Batch Mechanism (`BatchMechanism`) connection option has been enhanced to support the native batch protocol for batch inserts. When BatchMechanism is set to 4 (NativeBatch), the driver uses the native batch protocol for the insert operation of all batched parameters. See Batch Mechanism on page 94 for details.
  - The Trust Store (`Truststore`) connection option has been enhanced and a new pre-connection attribute, `SQL_COPT_INMEMORY_TRUSTSTORECERT`, has been added to support specifying the contents of the TLS/SSL certificates for TLS/SSL server authentication. Specifying certificate content directly eliminates the need to store the truststore file on the disk and lets applications use TLS/SSL server authentication without any disk dependency. See Trust Store on page 134 and Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 66 for details.
  - The Use Declare Fetch (`UseDeclareFetch`) and Fetch Size (`FetchSize`) connection options have been added to the driver. The Use Declare Fetch option determines whether the driver attempts to return a result set in a single fetch or across multiple fetches. If multiple fetches are used, the setting of the Fetch Size option determines the size of each fetch. For large result sets, retrieving rows in multiple fetches can improve response time and thereby prevent possible timeouts. See Use Declare Fetch on page 137 and Fetch Size on page 109 for details.

    Note: `B6` is an alias for the Use Declare Fetch connection option and `Fetch` and `A7` are the aliases for the Fetch Size connection option.

- **Changed Behavior**
  - The default values for the Unbounded Numeric Precision (`UnboundedNumericPrecision`) and Unbounded Numeric Scale (`UnboundedNumericScale`) connection options have been changed to `-1`. Now, by default, the driver will return the values from the unbounded NUMERIC columns without modifying their precision and scale. See Unbounded Numeric Precision on page 135 and Unbounded Numeric Scale on page 136 for details.

Changes for 8.0.2 GA

- **Driver Enhancements**
  - The driver has been enhanced to support stored procedures for PostgreSQL 11 and later.
  - The Call Escape Behavior (CallEscapeBehavior) connection option has been added to the driver. The driver determines whether to call a user-defined function or a stored procedure based on the value specified for this connection option. See Call Escape Behavior on page 95 for details.

- The driver has been enhanced to support certificate-based authentication. It uses TLS/SSL protocol for authentication and authenticates the client to the server if the TLS/SSL client certificate is issued by a trusted Certificate Authority (CA) and the Common Name (`cn`) attribute of the TLS/SSL client certificate matches the database user name. Two new connection options, Client SSL Certificate (ClientSSLCertificate) and Client SSL Key (ClientSSLKey), have been added to support this authentication method. See Certificate-based authentication on page 64 for details.
- 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 131 for details.
- The driver has been enhanced to support the scram-sha-256-plus authentication method, which uses channel binding for establishing a secure connection with PostgreSQL (v11.0 and higher).
- The driver has been enhanced with the new Batch Mechanism (BatchMechanism) connection option, which specifies the preferred mechanism for executing batch insert operations. By setting Batch Mechanism to 2 (MultiRowInsert) or 3 (Copy), the driver can achieve substantial performance gains when performing batch inserts. The default setting is BatchMechanism=3 (Copy). See Batch Mechanism on page 94 for details.
- The driver has been enhanced to support the following data types: Citext, Float, and Tinyint. See Data Types on page 27 for details.
- The drivers using base version B0649 and later have been enhanced to include timestamp in the internal packet logs by default. If you want to disable the timestamp logging in packet logs, set `PacketLoggingOptions=1`. The internal packet logging is not enabled by default. To enable it, set `EnablePacketLogging=1`.
- The Driver Manager for UNIX/Linux has been enhanced to support setting the Unicode encoding type for applications on a per connection basis. By passing a value for the SQL_ATTR_APP_UNICODE_TYPE attribute using SQLSetConnectAttr, your application can specify the encoding at connection. This allows your application to pass both UTF-8 and UTF-16 encoded strings with a single environment handle. The valid values for the SQL_ATTR_APP_UNICODE_TYPE attribute are SQL_DD_CP_UTF8 and SQL_DD_CP_UTF16. The default value is SQL_DD_CP_UTF8.
- The new AllowedOpenSSLVersions option determines which version of the OpenSSL library file the driver uses for data encryption when multiple versions are installed with the product. For example, when specifying a value of 1.1.1 (`AllowedOpenSSLVersions=1.1.1`) the driver uses the 1.1.1 version of the library that is installed with the driver.
- A Power BI connector is now included with the product package. You can use this connector to access your PostgreSQL data with Power BI. See Power BI (Windows only) on page 39 for details.
- The driver has been enhanced to support Select queries with parameterized arrays.
- The driver has been enhanced to support md5 and scram-sha-256 authentication methods.
- The driver has been enhanced to support materialized views and foreign tables.

- **Changed Behavior**
  - For PostgreSQL 9.0 and later, the driver behavior has been updated to support executing multiple prepared statements in a single query that contain inserts for BYTEA values. However, for versions earlier than PostgreSQL 9.0, this functionality is not supported and the driver returns an error.
  - The default value for the Batch Mechanism (BatchMechanism) connection option has been changed to 3 (Copy). The driver now, by default, uses the PostgreSQL COPY command to insert rows into the target table. It allows the driver to achieve substantial performance improvements over `1` (SingleRowInsert) when performing batch inserts. See Batch Mechanism on page 94 for details.
  - Support has been deprecated for HP-UX IPF and HP-UX PA-RISC platforms.

- The following Windows platforms have reached the end of their product lifecycle and are no longer supported by the driver:
  - Windows 8.0 (versions 8.1 and higher are still supported)
  - Windows Vista (all versions)
  - Windows XP (all versions)
  - Windows Server 2003 (all versions)

Driver requirements

Data source and platform requirements

Refer to Supported Configurations for the latest data source and platform requirements.

Windows requirements for 32-bit drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- An application that is compatible with components that were built using Microsoft Visual Studio 2015 compiler and the standard Win32 threading model.
- 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.
- An application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 14.00.40310.41 and the standard Windows 64 threading model.
- 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.

Installing and setting up the driver (Windows)

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

To begin accessing data with the driver:

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

   `PROGRESS_DATADIRECT_ODBC_`*`nn`*`_WIN_`*`xx`*`_INSTALL.exe`

   c) Follow the prompts to complete installation.

   Note:

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

   ---

2. To configure the driver using the ODBC Administrator (GUI), start the ODBC Administrator from the Progress DataDirect program group. The GUI dialog allows you to configure the data source definitions in the Windows Registry or generate connection strings.

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

3. Select either the **User DSN**, **System DSN**, or **File DSN** tab to display a list of data sources.
   - **User DSN**: If you installed a default DataDirect ODBC user data source as part of the installation, select the appropriate data source name and click **Configure** to display the driver Setup dialog box.

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: To configure a new system data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.
   - **File DSN**: To configure a new file data source, click **Add** to display a list of installed drivers. Select your 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.
4. On the **General** tab of the driver Setup dialog box, provide values for the following essential connection options for user ID/password (No Encryption) authentication; then, click **Apply**:
   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Host Name**: Type the name or the IP address of the server to which you want to connect.
   - **Port Number**: Type the port number of the server listener. The default is `5432`.
   - **Database Name**: Type the name of the database to which you want to connect.

Note: For information on other authentication methods, see Authentication on page 61.

5. Set the values for any additional connection options that you want to configure. To view more options, click on the tabs on the dialog. See the following resources for additional information on optional features and functionality:
   - Connection string examples on page 17 provides a connection string example that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
   - Connection option descriptions on page 83 provides a complete list of supported options by functionality.
   - Performance considerations on page 73 describes connection options that affect performance, along with recommended settings.
6. Click **Test Connect** to attempt to connect to the data source using the connection options.
7. The logon dialog appears. Update the following fields; then, click **OK**.
   - **User Name**: Type your user name as specified on the PostgreSQL server.
   - **Password**: Type your password.

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

8. If the test was successful, the window displays a confirmation message.
9. Click **OK** to close the Setup dialog. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the Setup dialog to modify your data source, or you can override these defaults by connecting to the data source using a connection string with alternate values.
10. Connect to your server and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:
    - Example Application: The example application allows you to test connect, execute SQL statements, and practice using the ODBC API right out of the box.
    - Power BI: Power BI is a business intelligence software program that allows you to generate analytics and visualized representations of your data.
    - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
    - Microsoft Excel: Excel is a spreadsheet tool that allows you to connect, view tables, and execute SQL statements against your data.

This completes the deployment of the driver.

See also

Using a connection string on page 54

Installing and setting up the driver (UNIX/Linux)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_`*`nn`*`_LINUX_`*`xx`*`_INSTALL.bin`

   c) Follow the prompts to complete installation.

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

a) Check your permissions. You must log in as a user with full r/w/x permissions recursively on the entire product installation directory.

b) Run one of the following product setup scripts from the installation directory to set variables: `odbc.sh` or `odbc.csh`. For Korn, Bourne, and equivalent shells, execute `odbc.sh`. For a C shell, execute `odbc.csh`. Executing the setup script:

c) Set the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For example, if you use an installation directory of `/opt/odbc` and the default system information file name, you would enter:

- **Korn or Bourne shell**: `ODBCINI=/opt/odbc/odbc.ini; export ODBCINI`
- **C shell**: `setenv ODBCINI /opt/odbc/odbc.ini`

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

- **odbc.ini file**: You can begin using the driver immediately by editing the `odbc.ini` file in the installation directory with a text editor. The following demonstrates a data source definition with the minimum attributes required for user ID/password authentication.

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

[PostgreSQL]
Driver=ODBCHOME/lib/xxpsql28.yy
...
HostName=PostgreSQL2
...
PortNumber=5432
...
Database=Payroll
...
UID=jsmith
...
Password=secret
...
```

  See Configuration through the system information (odbc.ini) file on page 55 for more information.

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

- **Connection string**: The driver also supports using connection strings for DSN (data source name), File DSN, or DSN-less connections. See Using a connection string on page 54, DSN-less connections, for more information. For examples, see Connection string examples on page 17.

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

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

- Connection string examples on page 17 provides a connection string example that can be used to configure common functionality and features. You can modify this example to create a string that best suites your environment.

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

- Connection option descriptions on page 83 provides a complete list of supported options by functionality.
- Performance considerations on page 73 describes connection options that affect performance, along with recommended settings.

5. Connect to your server and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resource guides you through accessing data:
   - Example Application: The example application is a command-line tool that allows you to test connect, execute SQL statements, and practice using the ODBC API in environments that do not support GUIs.

This completes the deployment of the driver.

Connection string examples

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

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

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

See also

Using a connection string on page 54

User ID/password (No Encryption) authentication

This string includes the options used to connect with basic user ID and password authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;HostName=host_name;PortNumber=port_number;
Database=database_name;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

where:

host_name

specifies the name or the IP address of the server to which you want to connect.

port_number

specifies the port number of the server listener. The default value is `5432`.

database_name

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

user_name

specifies your username.

password

specifies your password.

attribute=value

specifies connection option settings. Multiple connection option attributes are separated by a semi-colon.

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.

The following example connection string includes the options for connecting with the user ID/password authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;HostName=PostgreSQL2;PortNumber=5432;
Database=PAYROLL;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 54

Connection option descriptions on page 83

Kerberos authentication

This string includes the options used to connect with the Kerberos authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;AuthenticationMethod=4;HostName=host_name;
PortNumber=port_number;Database=database_name;GSSClient=gss_client_library;
ServicePrincipalName=service_principal_name;LogonID=user_name;
[attribute=value[;...]];
```

where:

host_name

specifies the name or the IP address of the server to which you want to connect.

port_number

specifies the port number of the server listener. The default value is `5432`.

database_name

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

gss_client_library

specifies the name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

service_principal_name

specifies the service principal name to be used by the driver.

user_name

specifies your username.

attribute=value

specifies connection option settings. Multiple connection option attributes are separated by a semi-colon.

The following example connection string includes the options for connecting with the Kerberos authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;AuthenticationMethod=4;HostName=myserver;PortNumber=5432;
Database=Payroll;GSSClient=gss123;ServicePrincipalName=postgres/myserver.example.com@EXAMPLE.COM;
LogonID=John;
```

See also

Using a connection string on page 54

Connection option descriptions on page 83

Certificate-based authentication

This string includes the options used to connect with certificate-based authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;LogonID=username;AuthenticationMethod=18;
Truststore=truststorename;TruststorePassword=truststorepassword;
ClientSSLCertificate=clientsslcertificate;ClientSSLKey=clientsslkey;
HostNameInCertificate=hostnameincertificate;
```

where:

username

specifies your username.

truststorename

specifies the directory that contains the truststore file. The truststore file contains a list of the valid CAs that are trusted by the client machine for TLS/SSL server authentication.

truststorepassword

specifies the password that is used to access the truststore file.

clientsslcertificate

specifies the full path of the certificate required to authenticate the client to the server.

clientsslkey

specifies the key used to access the client SSL certificate.

hostnameincertificate

specifies a host name for certificate validation.

The following example connection string includes the options for connecting with certificate-based authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;LogonID=odbc0123;
AuthenticationMethod=18;Truststore=TrustStoreName;TruststorePassword=TSXYZZY;
ClientSSLCertificate=C:\abc\odbc0123.crt;ClientSSLKey=C:\abc\odbc0123.key;
HostNameInCertificate=MySubjectAltName;
```

See also

Using a connection string on page 54
Connection option descriptions on page 83
Certificate-based authentication on page 64

Connection Failover

This string configures the driver to use connection failover in conjunction with some of its optional features. It uses the user ID/password (No Encryption) authentication method for authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;
AlternateServers=alternate_server;ConnectionRetryCount=connection_retry_count;
ConnectionRetryDelay=connection_retry_delay;LoadBalancing=load_balancing;
FailoverMode=failover_mode;LogonID=user_name;Password=password;
```

where:

alternate_servers

specifies addresses of the alternate database servers to which the driver tries to connect if the primary database server is unavailable.

connection_retry_count

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

connection_retry_delay

specifies the number of seconds the driver waits between connection retry attempts.

load_balancing

determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. See "Load Balancing" for details.

failover_mode

specifies the type of failover method the driver uses. See "Failover Mode" for details.

user_name

specifies your username.

password

specifies your password.

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.

The following example connection string includes the options for configuring the driver for connection failover.

```
DRIVER=DataDirect 8.0 PostgreSQL;
AlternateServers=(HostName=postgreSQL:PortNumber=5432:Database=Accounting,
HostName=255.201.11.24:PortNumber=5431:Database=Accounting);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0;
LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 54
Connection option descriptions on page 83
Failover support on page 69
Load Balancing on page 119
Failover Mode on page 107

TLS/SSL client authentication

This string configures the driver to use the TLS/SSL client authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate. In addition, it uses user ID and password for authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;EncryptionMethod=1;
HostName=host_name;HostNameInCertificate=hostnameincertificate;
PortNumber=port_number;Database=database_name;Keystore=keystore_name;
KeystorePassword=keystore_password;ValidateServerCertificate=validate_server_certificate;
LogonID=user_name;Password=password;
```

where:

`host_name`

specifies the name or the IP address of the server to which you want to connect.

`hostnameincertificate`

specifies a host name for certificate validation.

`port_number`

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

`database_name`

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

`keystore_name`

specifies the name of the directory containing the keystore file to be used.

`keystore_password`

specifies the password used to access the keystore file.

`validate_server_certificate`

determines whether the driver validates the certificate that is sent by the database server. See "Validate Server Certificate" for details.

`user_name`

specifies your username.

`password`

specifies your password.

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.

The following example connection string includes the options for connecting with the TLS/SSL client authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;EncryptionMethod=1;
HostName=YourServer;HostNameInCertificate=MySubjectAltName;
PortNumber=5432;Database=PAYROLL;Keystore=KeyStoreName;KeystorePassword=YourKSPassword;
ValidateServerCertificate=1;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 54

Connection option descriptions on page 83

TLS/SSL encryption on page 65

Validate Server Certificate on page 138

TLS/SSL server authentication

This string configures the driver to use the TLS/SSL server authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate. In addition, it uses user ID and password for authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;EncryptionMethod=1;HostName=host_name;
HostNameInCertificate=hostnameincertificate;PortNumber=port_number;Database=database_name;
Truststore=truststore;TruststorePassword=truststore_password;
ValidateServerCertificate=validate_server_certificate;
LogonID=user_name;Password=password;
```

where:

host_name

specifies the name or the IP address of the server to which you want to connect.

hostnameincertificate

specifies a host name for certificate validation.

port_number

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

database_name

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

truststore

specifies either the path and file name of the truststore file or the contents of the TLS/SSL certificates to be used.

When specifying the contents of the TLS/SSL certificates, use the following format:

`Truststore=data://-----BEGIN CERTIFICATE-----`*`certificate_content`*`-----END CERTIFICATE-----`.

Where *`certificate_content`* is the content of the TLS/SSL certificate. Note that the number of dashes (`-----`) must be the same before and after both `BEGIN CERTIFICATE` and `END CERTIFICATE`.

`truststore_password`

specifies the password that is used to access the truststore file.

Note: Do not specify the password when using the certificate content for authentication. Since the truststore file is not required to be stored on the disk when the certificate content is specified directly, the driver need not unlock its contents.

`validate_server_certificate`

determines whether the driver validates the certificate that is sent by the database server. See "Validate Server Certificate" for details.

`user_name`

specifies your username.

`password`

specifies your password.

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.

The following example connection string includes the options for connecting with the TLS/SSL server authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;EncryptionMethod=1;HostName=YourServer;
HostNameInCertificate=MySubjectAltName;PortNumber=5432;Database=PAYROLL;
Truststore=TrustStoreFile;TruststorePassword=TSXYZZY;
ValidateServerCertificate=1;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 54
Connection option descriptions on page 83
TLS/SSL encryption on page 65
Validate Server Certificate on page 138

Proxy server

This string includes the options you may need to connect through a proxy server with basic user ID and password authentication.

Note: The strings demonstrated in this section use the DSN-less format. For additional formats, see "Using a connection string".

```
DRIVER=DataDirect 8.0 PostgreSQL;
HostName=host_name;ProxyHost=proxy_host;ProxyPassword=proxy_password;
ProxyPort=proxy_port;ProxyUser=proxy_user;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

where:

host_name

specifies the name or the IP address of the server to which you want to connect.

proxy_host

specifies the proxy server to use for the first connection.

proxy_password

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

proxy_port

specifies the port number where the proxy server is listening for requests for the first connection. The default is `0`, which means the port number is determined by the setting of the Proxy Host (ProxyHost) option.

proxy_user

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

user_name

specifies your username.

password

specifies your password.

attribute=value

specifies connection option settings. Multiple connection option attributes are separated by a semi-colon.

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.

The following example connection string includes the options required for using a proxy server with the user ID/password authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;HostName=yourserver;
ProxyHost=pserver;ProxyPassword=proxys3cr3t;ProxyPort=1234;
ProxyUsert=jsmith;LogonID=jsmith@abc.com;Password=secret;
```

See also

Using a connection string on page 54

Connection option descriptions on page 83

Version string information

The driver has a version string of the format:

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

or

```
XX.YY.ZZZZ(bAAAA, 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 base component.

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

For example:

```
08.00.0002 (b0001, u0002)
      |__|  |___|  |___|
     Driver  Base  Utility
```

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.

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 ivpsql28.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: On AIX, Linux, and Solaris, the full path to the driver does not have to be specified for the test loading tool.

getFileVersionString function

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

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

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

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

Data Types

The following table shows how the PostgreSQL data types are mapped to the standard ODBC data types. Using the XML Data Type on page 31 describes PostgreSQL to Unicode data type mappings.

Table 1: PostgreSQL Data Types

PostgreSQL	ODBC
Bigint	SQL_BIGINT
Bigserial	SQL_BIGINT
Bit[1]	SQL_BIT
Bit varying	SQL_VARBINARY
Boolean	SQL_BIT
Bytea	SQL_VARBINARY
Character	SQL_CHAR
Character varying	SQL_VARCHAR
Citext[2,3]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double Precision	SQL_DOUBLE
Float	SQL_REAL
Integer	SQL_INTEGER
Money	SQL_VARCHAR
Name	SQL_VARCHAR
Numeric[4]	SQL_NUMERIC
Real	SQL_REAL
Serial	SQL_INTEGER
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR
Time[5]	SQL_TYPE_TIME

[1] Bit maps to SQL_BIT when the length for the bit is 1. If the length is greater than 1, the driver maps the column to SQL_BINARY.
[2] The Citext data type behaves the same as the Text data type, except that it is case-insensitive. The select operations performed on Citext columns return case-insensitive results.
[3] Supported for PostgreSQL versions 9.4 and higher.
[4] Numeric maps to SQL_NUMERIC if the precision of the Numeric is less than or equal to 38. If the precision is greater than 38, the driver maps the column to SQL_VARCHAR.
[5] Time mapping changes based on the setting of the Fetch TWFS as Time option

PostgreSQL	ODBC
Timestamp	SQL_TYPE_TIMESTAMP
Timestamp with timezone[6]	SQL_VARCHAR
Tinyint	SQL_SMALLINT
Wchar	SQL_CHAR
Wvarchar	SQL_VARCHAR
XML	SQL_WLONGVARCHAR

See Retrieving Data Type Information on page 31 for more information about data types.

Partially and Unsupported Data Types

The following section describes how the driver handles partially and unsupported data types.

Partially supported data types

The following table contains the PostgreSQL data types that are partially supported by the driver. These data types map the SQL_VARCHAR ODBC type with a scale of 0.

Table 2: Partially Supported Data Types

PostgreSQL Data Type	Precision
Box	255
Cidr	50
Inet	50
Int4	255
Interval	255
Lseg	255
Macaddr	17
Money	255
Point	255
Point	255

[6] Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.

PostgreSQL Data Type	Precision
Polygon	255
Timetz	21

Unsupported data types

The following data types are not supported by the driver. When encountered by the driver, unsupported types map to the SQL_VARCHAR type with a precision of 10485760 and scale 0.

- UUID
- Record

Interval Types

The following Interval Types are not supported and map to SQL_VARCHAR with an incorrect precision:

- INTERVAL DAY
- INTERVAL DAY TO HOUR
- INTERVAL DAY TO MINUTE
- INTERVAL DAY TO SECOND
- INTERVAL HOUR
- INTERVAL HOUR TO SECOND
- INTERVAL MINUTE
- INTERVAL MINUTE TO SECOND
- INTERVAL MONTH
- INTERVAL SECOND
- INTERVAL YEAR
- INTERVAL YEAR TO MONTH

Information Schema Types

The columns of the information schema views use special data types that are defined in the information schema. These data types are not used while creating the table and are not supported by the driver. By default, the driver maps the following information schema types to SQL_VARCHAR:

- cardinal_number
- character_data
- sql_identifier
- time_stamp
- yes_or_no

Using the XML Data Type

By default, PostgreSQL returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the option XMLDescribeType (XDT) to SQL_LONGVARBINARY (-4) and bind the data as SQL_C_BINARY.

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.

On UNIX, Linux, or Windows, 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 long    columnSize;

    SQLINTEGER       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);
    }
}
```

Driver specifications

This section describes the general functionality supported by the driver.

- **ODBC Compliance**: 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 a list of supported API functions.

- **Unicode support**: The driver automatically determines whether the PostgreSQL database is a Unicode database.

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

- **Isolation and lock levels**: The driver supports isolation levels 0 (read uncommitted), 1 (read committed), 2 (repeatable read), and 3 (serializable). The default is 1. In addition, it supports record-level locking.

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

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

Additional information

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

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

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 2021, Release 8.0.2 for the Progress DataDirect for ODBC for PostgreSQL Wire Protocol Driver, Version 0001

2

Tutorials

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

For details, see the following topics:

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

The Example application

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

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

   A command prompt opens.

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

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

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

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

Tableau (Windows only)

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

To use the driver to access data with Tableau:

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

   ```
   DataDirect PostgreSQL.tdc
   ```

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

   `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. In the DSN field, select the data source you want to use from the drop-down menu. For example, **My DSN**. Then, click **Connect**. The Logon dialog appears pre-populated with the connection information you provided in your data source.
6. If required, type your user name and password; then, click **OK**. The Logon dialog closes. Then, click **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.

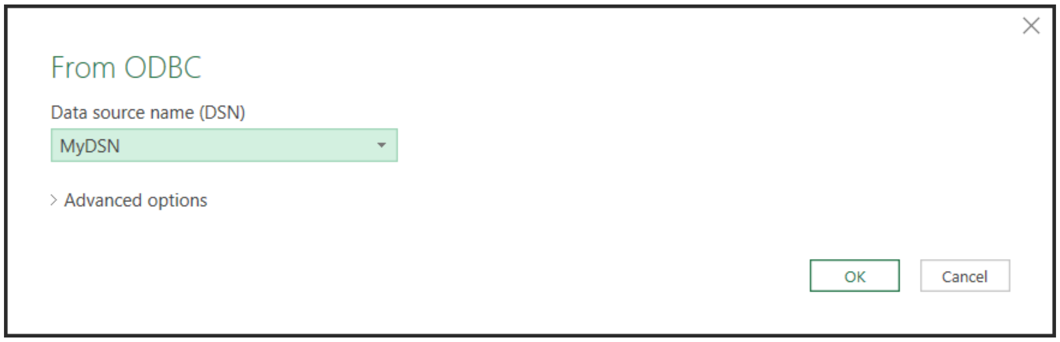
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

Microsoft Excel (Windows only)

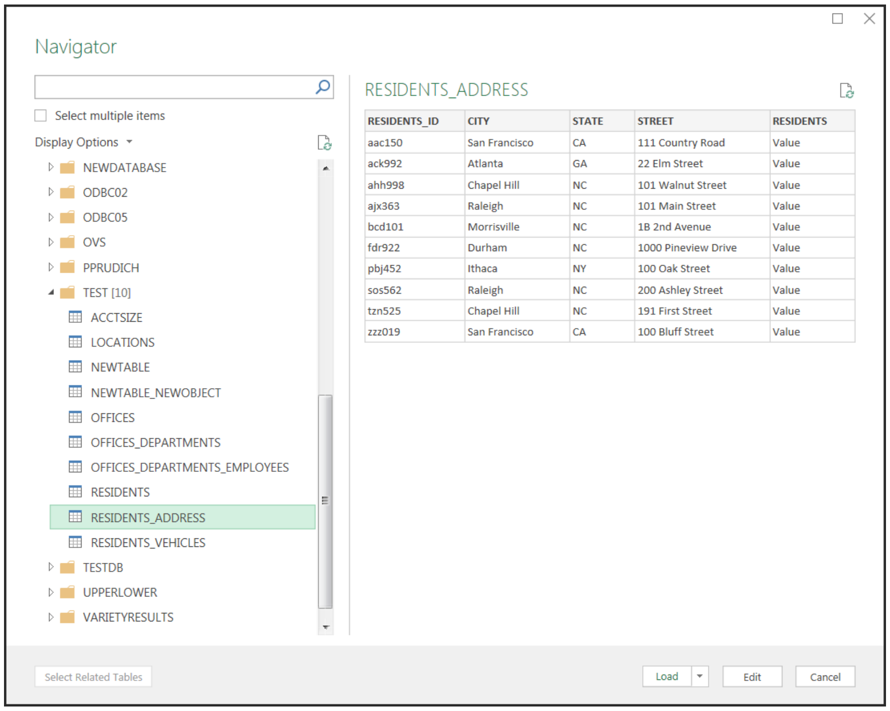
After you have configured your data source, you can use the driver to access your data with Microsoft Excel from the Data Connection Wizard. Using the driver with Excel provides improved performance when retrieving data, while leveraging the driver's relational-mapping tools.

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

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

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

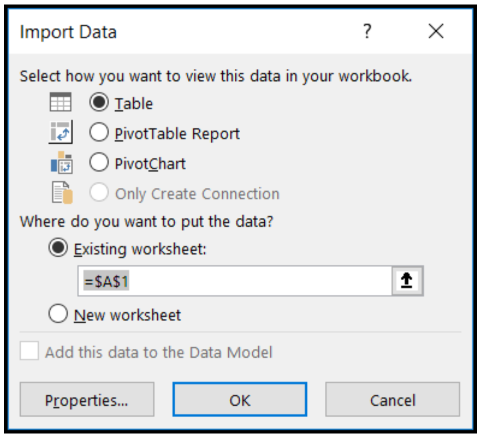


RESIDENTS_ADDRESS

RESIDENTS_ID	CITY	STATE	STREET	RESIDENTS
aac150	San Francisco	CA	111 Country Road	Value
ack992	Atlanta	GA	22 Elm Street	Value
ahh998	Chapel Hill	NC	101 Walnut Street	Value
ajx363	Raleigh	NC	101 Main Street	Value
bcd101	Morrisville	NC	1B 2nd Avenue	Value
fdr922	Durham	NC	1000 Pineview Drive	Value
pbj452	Ithaca	NY	100 Oak Street	Value
sos562	Raleigh	NC	200 Ashley Street	Value
tzn525	Chapel Hill	NC	191 First Street	Value
zzz019	San Francisco	CA	100 Bluff Street	Value

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

6. Load your data:
   - Click **Load** to import your data into your work sheet. Skip to the end.
   - Click **Load**>**Load To** to specify a location to import your data. Proceed to the next step.
7. The **Import Data** window appears.



Select 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. For more information, refer to the Microsoft Excel product documentation at: https://support.office.com/.

Power BI (Windows only)

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

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

     ```
     %USERPROFILE%\Documents\Power BI Desktop\Custom Connectors
     ```
   - The following Windows registry entry is updated.

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

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

3

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. 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 all platforms, file data sources, which are simply text files, can be stored locally or on a network computer, and are accessible to other machines. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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

For details, see the following topics:

- Environment settings
- Configuring the driver using the GUI
- Using a connection string
- Additional configuration methods for UNIX and Linux
- Using a logon dialog box
- Authentication
- TLS/SSL encryption
- Failover support

- DataDirect connection pooling
- Performance considerations

Environment settings

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

UNIX/Linux environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

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

- `LD_LIBRARY_PATH` on Linux and Oracle Solaris
- `LIBPATH` on AIX

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

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

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

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

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

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

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

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

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

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

ODBCINST

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

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

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

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

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

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

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

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

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

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.

The Test Loading Tool

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

Then 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.

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

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

Note: On Solaris, AIX, and Linux, the full path to the driver does not have to be specified for the tool.

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

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

UTF-16 applications on 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);
  ```

Configuring the driver using the 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.

On UNIX and Linux, data sources are stored in the `odbc.ini` file. In addition to manually editing the file, you can configure and modify data sources through the UNIX/Linux 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 PostgreSQL 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. On the General tab, specify values for the following options:
   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Host Name**: Type the name or the IP address of the server to which you want to connect.
   - **Port Number**: Type the port number of the server listener. The default is `5432`.

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

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. If applicable, on the Advanced tab, provide values for options to configure advanced behavior.
6. If applicable, on the Security tab, configure security settings.
7. If applicable, on the Failover tab, configure failover data source settings.
8. If applicable, on the Pooling tab, configure connection pooling settings.
9. 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.

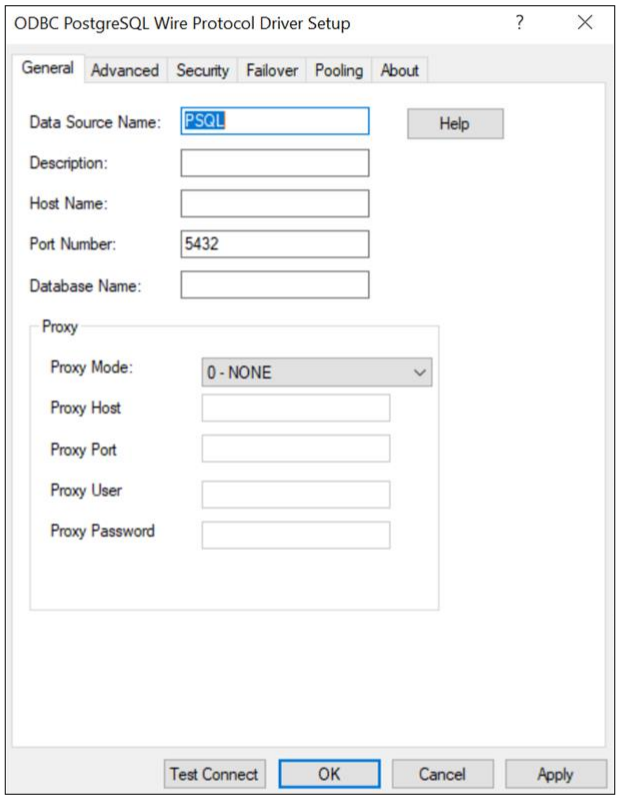
See also

Using a logon dialog box on page 61

General tab

The General tab allows you to configure essential and required options that are used to create a data source. The fields are optional unless otherwise noted.

Figure 1: General tab



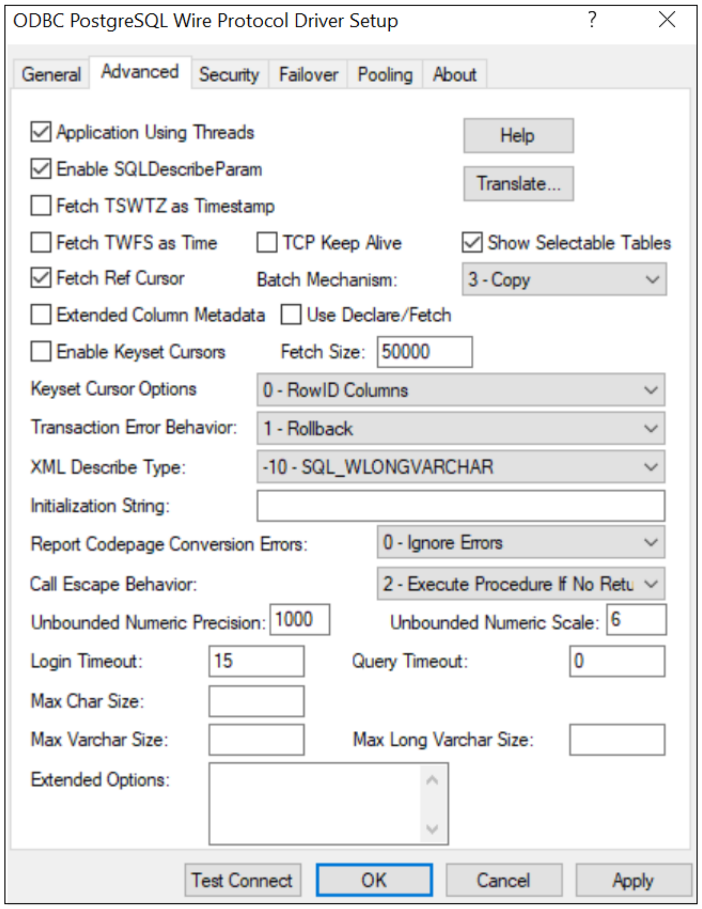
Data Source Name: PSQL
Description:
Host Name:
Port Number: 5432
Database Name:
Proxy
Proxy Mode: 0 - NONE
Proxy Host
Proxy Port
Proxy User
Proxy Password

Connection Options: General	Default
Data Source Name on page 101	No default value
Description on page 102	No default value
Host Name on page 112	No default value
Port Number on page 124	`5432`
Database Name on page 102	No default value
Proxy Mode on page 126	`0` (NONE)
Proxy Host on page 125	No default value
Proxy Port on page 127	`0`
Proxy User on page 128	No default value
Proxy Password on page 126	No default value

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


ODBC PostgreSQL Wire Protocol Driver Setup

General | Advanced | Security | Failover | Pooling | About

[✓ Application Using Threads]
[✓ Enable SQLDescribeParam]
Fetch TSWTZ as Timestamp
Fetch TWFS as Time
TCP Keep Alive
[✓ Show Selectable Tables]
[✓ Fetch Ref Cursor]
Batch Mechanism: 3 - Copy
Extended Column Metadata
Use Declare/Fetch
Enable Keyset Cursors
Fetch Size: 50000
Keyset Cursor Options: 0 - RowID Columns
Transaction Error Behavior: 1 - Rollback
XML Describe Type: -10 - SQL_WLONGVARCHAR
Initialization String:
Report Codepage Conversion Errors: 0 - Ignore Errors
Call Escape Behavior: 2 - Execute Procedure If No Retu
Unbounded Numeric Precision: 1000
Unbounded Numeric Scale: 6
Login Timeout: 15
Query Timeout: 0
Max Char Size:
Max Varchar Size:
Max Long Varchar Size:
Extended Options:

Connection Options: Advanced	Default
Application Using Threads on page 92	Enabled
Enable SQLDescribeParam on page 104	Disabled
Fetch TSWTZ as Timestamp on page 110	Disabled
Fetch TWFS as Time on page 111	Disabled
TCP Keep Alive on page 132	Disabled
Show Selectable Tables on page 131	Enabled
Fetch Ref Cursors on page 109	Enabled
Batch Mechanism on page 94	`3-Copy`

Connection Options: Advanced	Default
Extended Column MetaData on page 105	Disabled
Use Declare Fetch on page 137	Disabled
Enable Keyset Cursors on page 103	Disabled
Fetch Size on page 109	`50000`
Keyset Cursor Options on page 117	`0-RowID Columns`
Transaction Error Behavior on page 133	`1-Rollback`
XML Describe Type on page 139	`-10`
Initialization String on page 115	No default value
Report Codepage Conversion Errors on page 129	`0-Ignore Errors`
Call Escape Behavior on page 95	`2`
Unbounded Numeric Precision on page 135	`-1`
Unbounded Numeric Scale on page 136	`-1`
Login Timeout on page 120	`15`
Query Timeout on page 129	`0`
Max Char Size on page 120	No default value
Max Varchar Size on page 122	No default value
Max Long Varchar Size on page 121	No default value
Extended Options on page 106	No default value

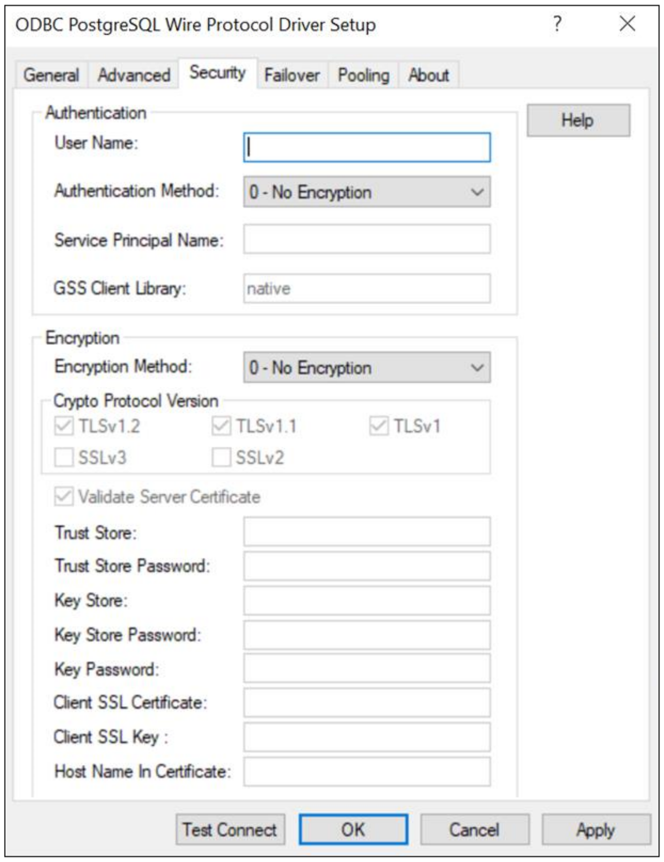
See also

Configuring the driver using the 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**.

Figure 3: Security tab


ODBC PostgreSQL Wire Protocol Driver Setup

General | Advanced | Security | Failover | Pooling | About

Authentication

User Name:

Authentication Method: 0 - No Encryption

Service Principal Name:

GSS Client Library: native

Encryption

Encryption Method: 0 - No Encryption

Crypto Protocol Version

- [x] TLSv1.2
- [x] TLSv1.1
- [x] TLSv1
- [ ] SSLv3
- [ ] SSLv2

- [x] Validate Server Certificate

Trust Store:

Trust Store Password:

Key Store:

Key Store Password:

Key Password:

Client SSL Certificate:

Client SSL Key :

Host Name In Certificate:

Table 3: Security Tab Connection Options

Connection Options: Security	Default
User Name on page 138	No default value
Authentication Method on page 93	`0-No Encryption`
Service Principal Name on page 130	No default value
GSS Client Library on page 112	`native`
Encryption Method on page 104	`0-No Encryption`
Crypto Protocol Version on page 100	`TLSv1.2,TLSv1.1,TLSv1`
Validate Server Certificate on page 138	Enabled
Trust Store on page 134	No default value

Connection Options: Security	Default
Trust Store Password on page 135	No default value
Key Store on page 116	No default value
Key Store Password on page 117	No default value
Key Password on page 116	No default value
Client SSL Certificate on page 95	No default value
Client SSL Key on page 96	No default value
Host Name In Certificate on page 113	No default value

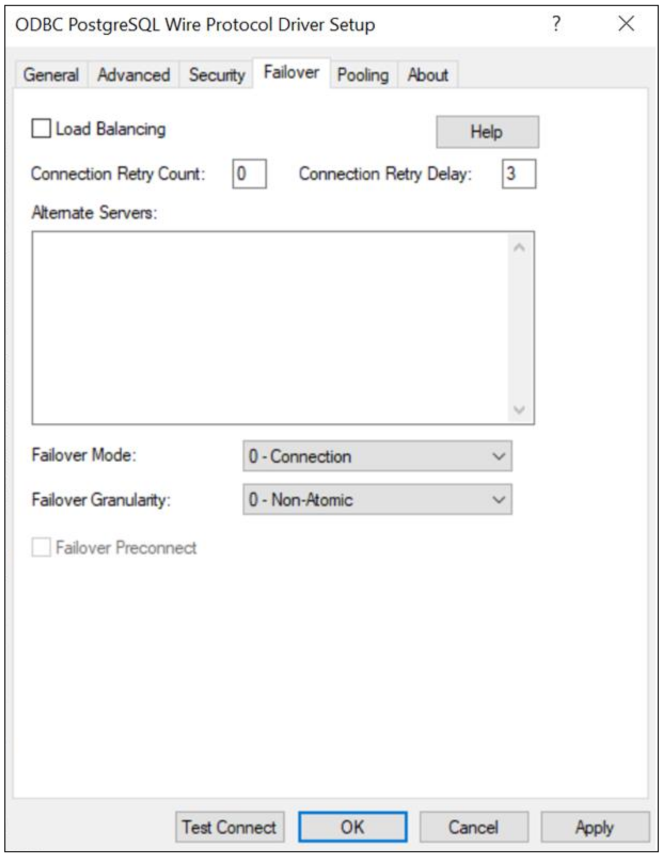
See also

TLS/SSL encryption on page 65

Failover tab

The Failover tab allows you to specify failover 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 4: Failover tab



Load Balancing [ ]
Connection Retry Count: 0
Connection Retry Delay: 3
Alternate Servers:
Failover Mode: 0 - Connection
Failover Granularity: 0 - Non-Atomic
Failover Preconnect [ ]

Connection Options: Failover	Default
Load Balancing on page 119	Disabled
Connection Retry Count on page 98	0
Connection Retry Delay on page 99	3
Alternate Servers on page 91	No default value
Failover Mode on page 107	**0 - Connection**
Failover Granularity on page 107	**0 - Non-Atomic**
Failover Preconnect on page 108	Disabled

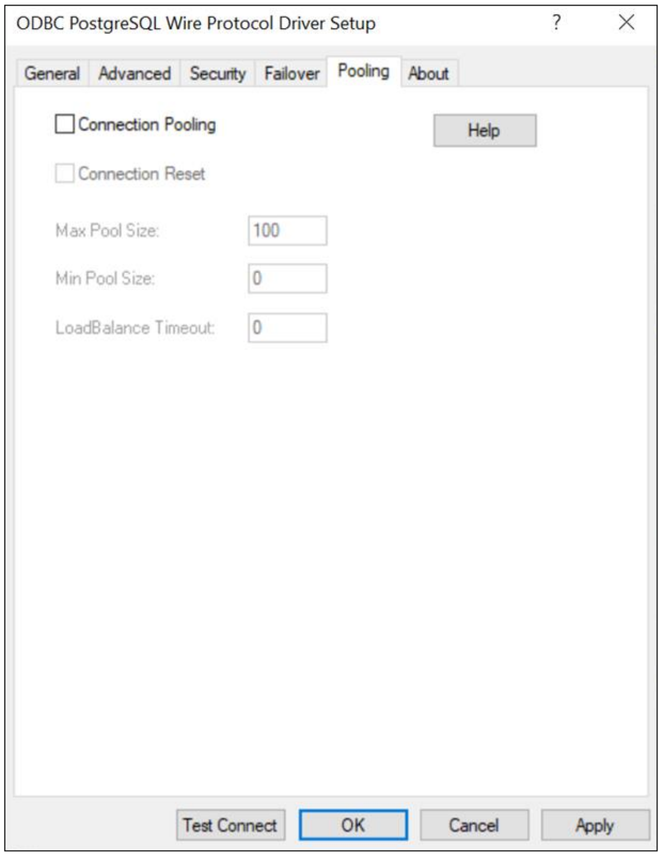
See also

Configuring the driver using the GUI on page 45

Pooling tab

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

Figure 5: Pooling tab



☐ Connection Pooling

☐ Connection Reset

Max Pool Size: 100

Min Pool Size: 0

LoadBalance Timeout: 0

Connection Options: Pooling	Default
Connection Pooling on page 96	Disabled
Connection Reset on page 97	Disabled
Max Pool Size on page 122	`100`
Min Pool Size on page 123	`0`
Load Balance Timeout on page 118	`0`

See also

Configuring the driver using the 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 driver for UNIX/Linux or Windows is:

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

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

```
FILEDSN=PostgreSQLWP.dsn;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect 8.0 PostgreSQL Wire Protocol;
HOST=PostgreSQLServer;PORT=5432;DB=Pgredb1;UID=JOHN;PWD=XYZZY
```

See also

Connection option descriptions on page 83
Connection string examples on page 17

Additional configuration methods for UNIX and Linux

This section contains configuration methods that are specific to the UNIX and Linux environments.

Configuration through the system information (odbc.ini) file

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.

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

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

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

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

```
PostgreSQL=DataDirect 8.0 PostgreSQL Driver
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 83
Sample default odbc.ini file on page 56
File data sources on page 59
IANAAppCodePage on page 114
DSN-less connections on page 58

Sample default odbc.ini file

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

```
[ODBC Data Sources]
PostgreSQL=DataDirect 8.0 PostgreSQL Driver

[PostgreSQL]
Driver=ODBCHOME/lib/ivpsql28.so
Description=DataDirect 8.0 PostgreSQL Driver
AllowedOpenSSLVersions=1.1.1,1.0.2
AlternateServers=
ApplicationUsingThreads=1
AuthenticationMethod=0
BatchMechanism=3
CallEscapeBehavior=2
ClientSSLCertificate=
ClientSSLKey=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CryptoLibName=
CryptoProtocolVersion=TLSv1.2, TLSv1.1, TLSv1
Database=
EnableDescribeParam=1
EnableKeysetCursors=0
EncryptionMethod=0
```

```
ExtendedColumnMetaData=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchRefCursors=1
FetchSize=50000
FetchTSWTZasTimestamp=0
FetchTWFSasTime=0
GSSClient=native
HostName=
HostNameInCertificate=
InitializationString=
KeepAlive=0
KeyPassword=
Keystore=
KeystorePassword=
KeysetCursorOptions=0
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxCharSize=
MaxLongVarcharSize=
MaxPoolSize=100
MaxVarcharSize=
MinPoolSize=0
Password=
Pooling=0
PortNumber=5432
ProxyHost=
ProxyMode=0
ProxyPassword=
ProxyPort=0
ProxyUser=
QueryTimeout=0
ReportCodePageConversionErrors=0
ServicePrincipalName=
ShowSelectableTables=1
SSLLibName=
TransactionErrorBehavior=1
TrustStore=
TrustStorePassword=
UnboundedNumericPrecision=-1
UnboundedNumericScale=-1
UseDeclareFetch=0
ValidateServerCertificate=1
XMLDescribeType=-10

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

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

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

- **To modify a data source:**
  a) Using a text editor, open the `odbc.ini` file.
  b) Modify the default values for attributes in the data source definitions as necessary based on your system specifics.

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 83

DSN-less connections

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

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

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

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

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

```
DataDirect 8.0 PostgreSQL Driver=Installed
```

This section also includes additional information for each driver.

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

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

See also

ODBCINST on page 43

Configuration through the system information (odbc.ini) file on page 55

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 PostgreSQL Driver=Installed

[DataDirect 8.0 PostgreSQL Driver]
Driver=ODBCHOME/lib/ivpsql28.so
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
SQLLevel=0
UsageCount=1

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

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

File data sources

The Driver Manager on 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, Windows, UNIX, or Linux. See "Configuring and connecting to data sources" for a general description of ODBC data sources on Windows, UNIX, and Linux platforms.

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 PostgreSQL Driver
...
HostName=PostgreSQL2
...
PortNumber=5432
...
Database=Payroll
...
LogonID=jsmith
...
Password=secret
...
```

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.

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\PostgreSQL2.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/PostgreSQL2.dsn
```

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

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

```
FILEDSN=/home/users/john/filedsn/PostgreSQL2.dsn;UID=john;PWD=test01
```

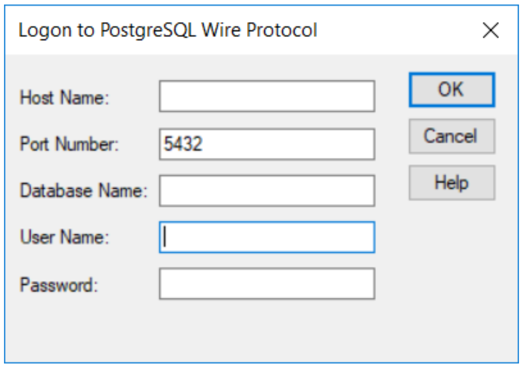
See also

Configuring and connecting to data sources on page 41
DSN-less connections on page 58
Additional configuration methods for UNIX and Linux on page 54

Using a logon dialog box

Some ODBC applications display a logon dialog box when you are connecting to a data source.

Figure 6: Logon to PostgreSQL dialog box



Host Name:
Port Number: 5432
Database Name:
User Name:
Password:

In this dialog box, provide the following information:

1. In the Host Name field, type either 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. The default is `5432`.
3. In the Database Name field, type the name of the database to which you want to connect.
4. In the User Name field, type your PostgreSQL user name.
5. In the Password field, type your password.
6. Click **OK** to complete the logon.

Authentication

The driver supports the following authentication methods:

- *User ID/password (No Encryption) authentication* authenticates using the specified user IDs and passwords.
- *Kerberos authentication* is a trusted third-party authentication service that verifies user identities. The driver supports both Windows Active Directory Kerberos and MIT Kerberos implementations.
- *Certificate-based authentication* uses TLS/SSL protocol for authentication and authenticates the client to the server if the TLS/SSL client certificate is issued by a trusted Certificate Authority (CA) and the Common Name (`cn`) attribute of the TLS/SSL client certificate matches the database user name.

By default, the driver is configured to use user ID/password authentication (`AuthenticationMethod=0`).

See also

Authentication Method on page 93

User ID/password (No Encryption) authentication

The driver supports the *User ID/password authentication*. It authenticates the user to the database using a user name and password.

To configure the driver to use user ID/password 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 `5432`.
- 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.

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

Connection string

```
DRIVER=DataDirect 8.0 PostgreSQL;HostName=myserver;PortNumber=5432;
Database=Payroll;LogonID=John;Password=secret
```

odbc.ini

```
[PostgreSQL]
Driver=ODBCHOME/lib/xxpsql28.yy
...
HostName=myserver
...
PortNumber=5432
...
Database=Payroll
...
LogonID=John
...
Password=secret
...
```

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

Kerberos authentication

The driver supports the *Kerberos authentication*. Kerberos authentication can take advantage of the user name and password maintained by the operating system to authenticate users to the database or use another set of user credentials specified by the application.

The Kerberos method requires knowledge of how to configure your Kerberos environment. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

To use Kerberos authentication, the application user first must obtain a Kerberos Ticket Granting Ticket (TGT) from the Kerberos server. The Kerberos server verifies the identity of the user and controls access to services using the credentials contained in the TGT.

If the application uses Kerberos authentication from a Windows client, the application user does not explicitly need to obtain a TGT. Windows Active Directory automatically obtains a TGT for the user.

If the application uses Kerberos authentication from a UNIX or Linux client, the user must explicitly obtain a TGT. To obtain a TGT explicitly, the user must log onto the Kerberos server using the kinit command. For example, the following command requests a TGT from the server with a lifetime of 10 hours, which is renewable for 5 days:

```
kinit -l 10h -r 5d user
```

where *user* is the application user.

Refer to your Kerberos documentation for more information about using the kinit command and obtaining TGTs for users.

To configure the driver to use Kerberos authentication:

- Set the Authentication Method (AuthenticationMethod) option to 4.
- 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 5432.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect.
- Set the GSS Client Library (GSSClient) option to specify the name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).
- Set the Service Principal Name (ServicePrincipalName) option to specify the service principal name to be used by the driver.
- Set the User Name (LogonID) option to specify your user name.

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

Connection string

```
DRIVER=DataDirect 8.0 PostgreSQL;AuthenticationMethod=4;HostName=myserver;PortNumber=5432;
Database=Payroll;GSSClient=gss123;ServicePrincipalName=postgres/myserver.example.com@EXAMPLE.COM;
LogonID=John;
```

odbc.ini

```
[PostgreSQL]
Driver=ODBCHOME/lib/xxpsql28.yy
...
AuthenticationMethod=4
...
HostName=myserver
...
PortNumber=5432
...
Database=Payroll
...
GSSClient=gss123
...
ServicePrincipalName=postgres/myserver.example.com@EXAMPLE.COM
...
LogonID=John
...
```

Certificate-based authentication

The driver supports certificate-based authentication for PostgreSQL 11.0 and later. The certificate-based authentication performs both TLS/SSL server authentication and TLS/SSL client authentication. It does not require the client to specify the password. It authenticates the client to the server if the TLS/SSL client certificate is issued by a trusted Certificate Authority (CA) and the Common Name (`cn`) attribute of the TLS/SSL client certificate matches the database user name.

To configure the driver to use certificate-based authentication:

- Set the User Name (LogonID) option to specify your user name.
- Set the Authentication Method (AuthenticationMethod) option to either `17` (SSL Verify CA) or `18` (SSL Verify Full). When set to `17`, the driver verifies just the TLS/SSL client certificate, but when set to `18`, it verifies the host name along with the TLS/SSL client certificate.

  Note: When the Authentication Method option is set to either `17` or `18`, the SSL encryption is automatically enabled (`EncryptionMethod=1`).

- Set the Trust Store (TrustStore) option to specify the directory that contains the truststore file. The truststore file contains a list of the valid CAs that are trusted by the client machine for TLS/SSL server authentication.
- Set the Trust Store Password (TrustStorePassword) option to specify the password that is used to access the truststore file.
- Specify values for one of the following sets of options:
  - Client SSL Certificate and Client SSL Key:
    - Set the Client SSL Certificate (ClientSSLCertificate) option to specify the full path of the certificate required to authenticate the client to the server.
    - Set the Client SSL Key (ClientSSLKey) option to specify the key used to access the client SSL certificate.
  - Key Store and Key Store Password:
    - Set the Key Store (KeyStore) option to specify the full path of the client SSL certificate that is saved in `PFX` format.
    - Set the Key Store Password (KeyStorePassword) option to specify the password used to access the keystore file.
- Set the Host Name In Certificate (HostNameInCertificate) to specify the name or the IP address of the server to which you want to connect. If it is not specified, the driver uses the value specified for the Host Name option.

  Note: Specify a value for this option only when Authentication Method is set to `18`.

The following examples show the connection information required to establish a session using certificate-based authentication.

Connection string

```
DRIVER=DataDirect 8.0 PostgreSQL;LogonID=odbc0123;
AuthenticationMethod=18;Truststore=TrustStoreName;TruststorePassword=TSXYZZY;
ClientSSLCertificate=C:\abc\odbc0123.crt;ClientSSLKey=C:\abc\odbc0123.key;
HostNameInCertificate=MySubjectAltName;
```

odbc.ini

```
[PostgreSQL]
Driver=ODBCHOME/lib/xxpsql28.yy
...
LogonID=odbc0123;
...
AuthenticationMethod=18;
...
Truststore=TrustStoreName;
...
TruststorePassword=TSXYZZY;
...
ClientSSLCertificate=C:\abc\odbc0123.crt;
...
ClientSSLKey=C:\abc\odbc0123.key;
...
HostNameInCertificate=MySubjectAltName;
...
```

See also

Connection option descriptions on page 83

TLS/SSL encryption on page 65

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 driver supports the following protocols using OpenSSL cipher suites:

- TLS v1.2, TLS v1.1, TLS v1.0
- SSL v3, SSL v2

The encryption cipher suite defines the type of encryption that is used for any data exchanged through an 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

SSL 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 string attributes are available to the driver to provide this information: TrustStore and TrustStorePassword. The value of TrustStore is a pathname that specifies the location of the truststore file. The value of TrustStorePassword is the password required to access the contents of the truststore.

Note: To allow the client to use TLS/SSL server authentication without storing the truststore file on the disk, you can specify the contents of the TLS/SSL certificates using the Trust Store connection option. Alternatively, you can use the pre-connection attribute, `SQL_COPT_INMEMORY_TRUSTSTORECERT`, to specify the certificate content. For more information, see "Trust Store" and "Using SQL_COPT_INMEMORY_TRUSTSTORECERT".

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. ValidateServerCertificate, another connection string attribute, 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 string attribute, HostNameInCertificate, allows an additional method of server verification. When a value is specified for HostNameInCertificate, it must match the host name of the server, which has been established by the administrator. This prevents malicious intervention between the client and the server and ensures that the driver is connecting to the server that was requested.

See also

Trust Store on page 134
Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 66

Using SQL_COPT_INMEMORY_TRUSTSTORECERT

`SQL_COPT_INMEMORY_TRUSTSTORECERT` is a pre-connection attribute that specifies the contents of the TLS/SSL certificates for TLS/SSL server authentication. When using `SQL_COPT_INMEMORY_TRUSTSTORECERT`, the driver stores the certificate content in memory, which eliminates the need to store the truststore file on the disk and lets applications use TLS/SSL server authentication without any disk dependency.

Note: The certificate content can be specified using the Trust Store connection option as well. However, if it is specified using both Truststore and `SQL_COPT_INMEMORY_TRUSTSTORECERT`, `SQL_COPT_INMEMORY_TRUSTSTORECERT` takes precedence over Trust Store.

The following example shows how to specify the contents of 3 certificates using `SQL_COPT_INMEMORY_TRUSTSTORECERT`:

```
SQLCHAR certificate[] = "
-----BEGIN CERTIFICATE-----12345abc-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----abcd123456-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----aabbcc11----END CERTIFICATE-----";
//The content of each certificate must be specified between -----BEGIN CERTIFICATE-----
 and -----END CERTIFICATE-----. Also, the number of dashes (-----) must be the same
before and after both BEGIN CERTIFICATE and END CERTIFICATE.

...

SQLSetConnectAttr(dbc, SQL_COPT_INMEMORY_TRUSTSTORECERT, (SQLPOINTER)certificate,
SQL_IS_POINTER);

ret = SQLDriverConnect(dbc, NULL,
(SQLCHAR*)"DSN=PostgreSQLWP_SSL;UID=jsmith;PWD=secret", SQL_NTS,
NULL, 0, NULL, SQL_DRIVER_NOPROMPT);
```

See also

Trust Store on page 134

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.

The following examples show how to configure the driver to use data encryption via the SSL client authentication.

Connection string

This connection string configures the driver to use the TLS/SSL client authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate. In addition, it uses user ID and password for authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;EncryptionMethod=1;HostName=YourServer;
HostNameInCertificate=MySubjectAltName;PortNumber=5432;Database=PAYROLL;
Keystore=KeyStoreName;KeystorePassword=YourKSPassword;
ValidateServerCertificate=1;LogonID=John;Password=secret;
```

odbc.ini

This `odbc.ini` file configures the driver to use the TLS/SSL client authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by the HostNameInCertificate option. In addition, it uses user ID and password for authentication.

```
Driver=ODBCHOME/lib/ivpsqlxx.so
Description=DataDirect PostgreSQL Wire Protocol driver
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=5432
...
Database=PAYROLL;
...
KeyStore=KeyStoreName
...
KeystorePassword=KSXYZZY
...
ValidateServerCertificate=1
...
LogonID=John;
...
Password=secret;
...
```

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.

Designating an 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:

- Latest version: `xxtls28.dll`
- 1.0.2 and earlier versions: `xxssl28.dll`

UNIX/Linux:

- Latest version: `libxxtls28.so [.sl]`
- 1.0.2 and earlier versions: `libxxssl28.so [.sl]`

Designating a library in the default directory

If you are using the default directory structure for the product, you can use the AllowedOpenSSLVersions option to designate a library. To use the AllowedOpenSSLVersions option, specify the version number of the library you want to load. For example, `AllowedOpenSSLVersions=1.0.2` loads the 1.0.2 version of OpenSSL library using the following naming convention and format:

- Windows: `install_dir\drivers\xxssl28.so [.sl]`
- UNIX/Linux: `install_dir/lib/libxxtls28.so [.sl]`

Note that this method works only with OpenSSL library files that match Progress's naming convention and relative installation location.

If you are using the GUI, this option is not exposed on the setup dialog. Instead, use the Extended Options field on the Advanced tab to configure this option. For more information, see "AllowedOpenSSLVersions."

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

AllowedOpenSSLVersions on page 90
CryptoLibName on page 99
SSLLibName on page 131

Failover support

The driver supports the following failover methods:

- *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.

To support the failover feature and provide additional advantages related to it, the driver also supports:

- *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.
- *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.

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

See also

Configuring failover on page 70

Guidelines for primary and alternate servers on page 72

Configuring failover

To configure failover:

1. 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.

Connection string example

The following connection string configures the driver to use connection failover in conjunction with some of its optional features. It uses the user ID/password (No Encryption) authentication method for authentication.

```
DRIVER=DataDirect 8.0 PostgreSQL;
AlternateServers=(HostName=postgreSQL:PortNumber=5432:Database=Accounting,
HostName=255.201.11.24:PortNumber=5431:Database=Accounting);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0;
LogonID=John;Password=secret;
```

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 AcctPostgreSQL.

odbc.ini file example

To following example configures the 32-bit driver to use connection failover in conjunction with some of its optional features. It uses the user ID/password (No Encryption) authentication method for authentication.

```
Driver=ODBCHOME/lib/ivpsqlxx.so
Description=DataDirect PostgreSQL Wire Protocol driver
...
AlternateServers=(HostName=postgreSQL:PortNumber=5432:Database=Accounting,
HostName=255.201.11.24:PortNumber=5431:Database=Accounting)
...
ConnectionRetryCount=4
...
ConnectionRetryDelay=5
...
LoadBalancing=0
...
FailoverMode=1
...
FailoverPreconnect=1
...
LogonID=John;
...
Password=secret;
...
```

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.

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.

See also

Connection option descriptions on page 83

Guidelines for primary and alternate servers

To ensure that failover works correctly, alternate servers should mirror data on the primary server or be part of a configuration where multiple database nodes share the same physical data.

DataDirect connection pooling

Connection pooling allows you to *reuse* connections rather than creating a new one every time the driver needs to establish a connection to the underlying database. Your Progress DataDirect for ODBC driver enables connection pooling without requiring changes to your client application.

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

To configure the driver to use connection pooling:

- Set the Connection Pooling (Pooling) option to `1`.
- Set the Connection Reset (ConnectionReset) option to `1` or `0`. Setting it to `1` resets the state of connections removed from the connection pool for reuse by an application to the initial configuration of the connection. Setting it to `0` does not reset the state of connections.
- Set the Load Balance Timeout (LoadBalanceTimeout) option to specify an integer value to specify the amount of time, in seconds, to keep connections open in a connection pool.
- Set the Max Pool Size (MaxPoolSize) option to specify an integer value to specify the maximum number of connections within a single pool.
- Set the Min Pool Size (MinPoolSize) option to an integer value to specify the minimum number of connections that are opened and placed in a connection pool when it is created.
- 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 `5432`.
- 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.

The following examples show how to configure the driver to use connection pooling:

Connection string

```
DRIVER=DataDirect 8.0 PostgreSQL;Pooling=1;ConnectionReset=0;
LoadBalanceTimeout=0;MaxPoolSize=100;MinPoolSize=0;HostName=myserver;
PortNumber=5432;Database=Payroll;LogonID=John;Password=secret;
```

odbc.ini

```
[PostgreSQL]
Driver=ODBCHOME/lib/xxpsql28.yy
...
Pooling=1
...
ConnectionReset=0
...
LoadBalanceTimeout=0
...
MaxPoolSize=100
...
MinPoolSize=0
...
HostName=myserver;
...
PortNumber=5432;
...
Database=Payroll;
...
LogonID=John;
...
Password=secret;
...
```

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.

Performance considerations

The following connection options can enhance driver performance.

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.

Batch Mechanism (BatchMechanism): Setting BatchMechanism to `2` (MultiRowInsert), `3` (Copy), or `4` (NativeBatch) provides significant performance gains over `1` (SingleRowInsert) when executing batch inserts:

- When `BatchMechanism=2`, the driver executes a single insert statement for all the rows contained in a parameter array. If the size of the insert statement exceeds the available buffer memory of the server, the driver executes multiple statements.
- When `BatchMechanism=3`, the driver uses the PostgreSQL COPY command to insert rows into the target table.
- When `BatchMechanism=4`, the driver uses the PostgreSQL native batch protocol to insert all batched parameters.

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

- **Load Balance Timeout (LoadBalancing)**: 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.

Use Declare Fetch (UseDeclareFetch) and Fetch Size (FetchSize): Use Declare Fetch determines whether the driver attempts to return a result set in a single fetch or across multiple fetches. If multiple fetches are used, the size of the fetch is determined by the setting of the Fetch Size (FetchSize) option. For large result sets, retrieving rows in multiple fetches can improve response time and thereby prevent possible timeouts.

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

4

Additional features and functionality

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

For details, see the following topics:

Using IP addresses

The driver supports Internet Protocol (IP) addresses in the IPv4 and IPv6 formats.

If your network supports named servers, the server name specified in the data source can resolve to an IPv4 or IPv6 address.

In the following connection string example, the IP address for the PostgreSQL server is specified in IPv4 format:

```
DRIVER=DataDirect 8.0 PostgreSQL Wire Protocol Driver;
HOST=123.456.78.90;PORT=5432;
DB=PSQLACCT;UID=JOHN;PWD=XYZZYYou
```

In the following connection string example, the IP address for the PostgreSQL server is specified in IPv6 format:

```
DRIVER=DataDirect 8.0 PostgreSQL Wire Protocol Driver;
HOST=2001:DB8:0000:0000:8:800:200C:417A;PORT=5432;
DB=SQLSACCT;UID=JOHN;PWD=XYZZYYou
```

In addition to the normal IPv6 format, the drivers in the preceding tables support IPv6 alternative formats for compressed addresses. For example, the following connection string specifies the server using IPv6 format, but uses the compressed syntax for strings of zero bits:

```
DRIVER=DataDirect 8.0 PostgreSQL Wire Protocol Driver;
HOST=2001:DB8:0:0:8:800:200C:417A;PORT=5432;
DB=SQLSACCT;UID=JOHN;PWD=XYZZYYou
```

For complete information about IPv6 formats, go to the following URL:

http://tools.ietf.org/html/rfc4291#section-2.2

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.

User-defined function results

PostgreSQL provides functionality to create user-defined functions. PostgreSQL does not define a call mechanism for invoking a user-defined function. Instead, user-defined functions must be invoked via a SQL statement. For example, given a function defined as:

```
CREATE table foo (intcol int, varcharcol varchar(123))
CREATE or REPLACE FUNCTION insertFoo
  (IN idVal int, IN nameVal varchar) RETURNS void
  AS $$
    insert into foo values ($1, $2);
  $$
  LANGUAGE SQL;
```

must be invoked natively as:

```
SELECT * FROM insertFoo(100, 'Mark')
```

even though the function does not return a value or results. The Select SQL statement returns a result set that has one column named `insertFoo` and no row data.

The PostgreSQL Wire Protocol driver supports invoking user-defined functions using the ODBC call Escape. The previously described function can be invoked using:

```
{call insertFoo(100, 'Mark')}
```

PostgreSQL functions return data from functions as a result set. If multiple output parameters are specified, the values for the output parameters are returned as columns in the result set. For example, the function defined as:

```
CREATE or REPLACE FUNCTION addValues(in v1 int, in v2 int)
  RETURNS int
  AS $$
    SELECT $1 + $2;
  $$
  LANGUAGE SQL;
```

returns a result set with a single column of type SQL_INTEGER, whereas the function defined as:

```
CREATE or REPLACE FUNCTION selectFooRow2
  (IN idVal int, OUT id int, OUT name varchar)
  AS $$
    select intcol, varcharcol from foo where intcol = $1;
  $$
  LANGUAGE SQL
```

returns a result set that contains two columns, an id column, mapped to SQL_INTEGER, and a name column, mapped to SQL_VARCHAR.

In addition, when calling PostgreSQL functions that contain output parameters, the native syntax requires that the output parameter values be omitted from the function call. This, in addition to output parameter values being returned as a result set, makes the PostgreSQL behavior of calling functions different from most other databases.

The PostgreSQL Wire Protocol driver provides a mechanism that makes the invoking of functions more consistent with how other databases behave. In particular, the PostgreSQL Wire Protocol driver allows parameter markers for output parameters to be specified in the function argument list when the Escape call is used. The driver allows buffers to be bound to these output parameters. When the function is executed, the output parameters are removed from the argument list sent to the server. The driver extracts the output parameter values from the result set returned by the server and updates the bound output parameter buffers with those values. For example, the function `selectFooRow2` described previously can be invoked as:

```
sql = L"{call selectFooRow2(?, ?, ?)}";
retVal = SQLPrepare(hPrepStmt, sql, SQL_NTS);
retVal = SQLBindParameter(
  hPrepStmt, 1, SQL_PARAM_INPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf, 0, &idInd);
retVal = SQLBindParameter(
  hPrepStmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf2, 4, &idInd2);
retVal = SQLBindParameter(
  hPrepStmt, 3, SQL_PARAM_OUTPUT, SQL_C_WCHAR,
  SQL_VARCHAR, 30, 0, &nameBuf, 123, &nameInd);
retVal = SQLExecute(hPrepStmt);
```

The values of the id and name output parameters are returned in the `idBuf2` and `nameBuf` buffers.

If output parameters are bound to a function call, the driver returns the output parameters in the bound buffers. An error is returned if the number of output parameters bound when the function is executed is less than the number of output parameters defined in the function. If no output parameters are bound to a function call, the driver returns the output parameters as a result set.

PostgreSQL can also return results from a function as a refcursor. There can be, at most, one refcursor per result; however, a function can return multiple results where each result is a refcursor. A connection option defines how the driver handles refcursors. See Fetch Ref Cursors on page 109 for details about this option.

Persisting a result set as an XML data file

The 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
```

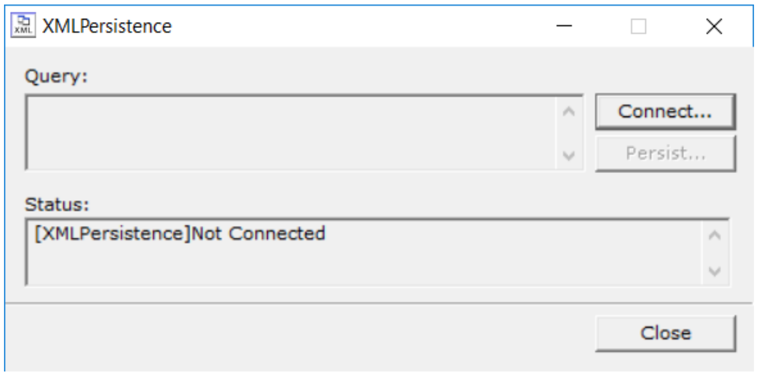
Using the Windows XML Persistence Demo tool

The driver for Windows is shipped with an XML persistence demo tool. This tool is installed in the product installation directory.

The tool has a graphical user interface and allows you to persist data as an XML data file.

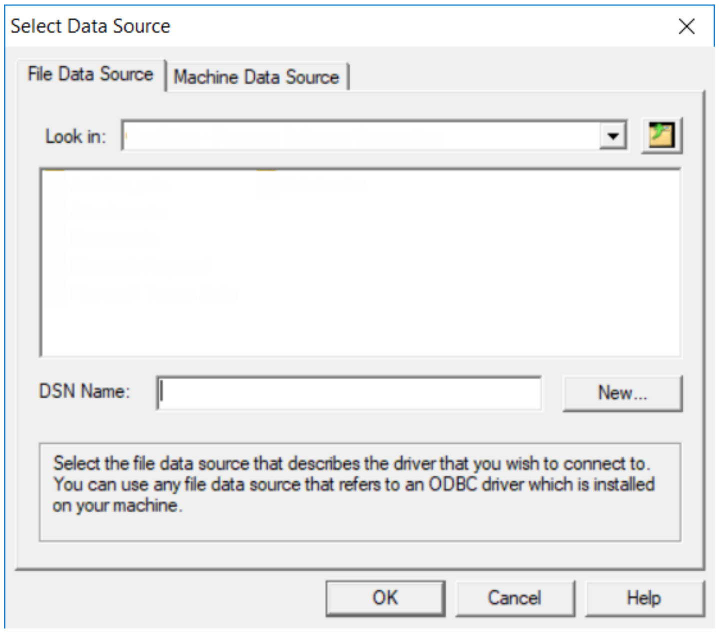
To use the Windows XML Persistence Demo tool:

1. From the product program group, select **XML Persistence Demo**. The XMLPersistence dialog box appears.



[XMLPersistence]Not Connected

2. First, you must connect to the database. Click **Connect**. The Select Data Source dialog box appears.



Select the file data source that describes the driver that you wish to connect to. You can use any file data source that refers to an ODBC driver which is installed on your machine.

3. You must either select an existing data source or create a new one. Take one of the following actions:
   - Select an existing data source and click **OK**.
   - Create a new file data source by clicking **New**. The Create New Data Source dialog box appears. Follow the instructions in the dialog box.
   - Create a new machine data source by clicking the **Machine Data Source** tab and clicking **New**. The Create New Data Source dialog box appears. Follow the instructions in the dialog box.

4. After you have connected to a database, type a SQL Select statement in the Query text box of the XML Persistence dialog box. Then, click **Persist**. The Save As dialog box appears.
5. Specify a name and location for the XML data file that will be created. Then, click **OK**.

   Note that the Status box in the XML Persistence dialog box displays whether the action failed or succeeded.

6. Click **Disconnect** to disconnect from the database.
7. Click **Close** to exit the tool.

Using the UNIX/Linux XML Persistence Demo tool

UNIX® On UNIX and Linux, the driver is shipped with an XML persistence demo tool named demoodbc. This tool is installed in the installation directory, in the `/samples/demo` subdirectory. For information about how to use this tool, refer to the demoodbc.txt file installed in the demo subdirectory.

Using Arrays of Parameters

PostgreSQL supports returning a set of output parameters or return values, but no ODBC standard method exists for returning arrays of output parameters or return values. If the call Escape is used to invoke a function that returns a set of output parameters and buffers are bound for those output parameters, the PostgreSQL Wire Protocol driver places the first set of output parameters in the bound buffers. If no output parameters are bound for functions that return a set of results or output parameters, the driver returns a result set with a row for each set of output parameters.

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.

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

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

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

The following tables provide a summary of supported connection options by functionality, including their attribute names, short names, and default values.

General options

User ID/password (No Encryption) options

Kerberos authentication options

TLS/SSL encryption options

Certificate-based authentication options

Failover options

Timeout options

Pooling options

General options

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

Table 4: General options

Attribute (Short Name)	Default
DataSourceName (DSN)	No default value
Database (DB)	No default value
Description (n/a)	No default value
HostName (HOST)	No default value
PortNumber (PORT)	`5432`

User ID/password (No Encryption) options

The following table summarizes options that are required for user ID/password authentication.

Table 5: User ID/password (No Encryption) options

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`0`
LoginID (UID)	No default value
Password (PWD)	No default value

Kerberos authentication options

The following table summarizes options that are required for Kerberos authentication.

Table 6: Kerberos authentication options

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`0`
GSSClient (GSSC)	`native`
ServicePrincipalName (SPN)	No default value

TLS/SSL encryption options

The following table summarizes the connection options used to enable TLS/SSL.

Table 7: TLS/SSL encryption options

Attribute (Short Name)	Default
AllowedOpenSSLVersions (AOV)	`1.1.1,1.0.2`
CryptoProtocolVersion (CPV)	`TLSv1.2,TLSv1.1,TLSv1`
CryptoLibName (CLN)	No default value
EncryptionMethod (EM)	`0` (No Encryption)
HostNameInCertificate (HNIC)	No default value
KeyPassword (KP)	No default value
Keystore (KS)	No default value
KeystorePassword (KSP)	No default value
SSLLibName (SLN)	No default value
Truststore (TS)	No default value
TruststorePassword (TSP)	No default value
ValidateServerCertificate (VSC)	`1` (Enabled)

Certificate-based authentication options

The following table summarizes the connection options used for certificate-based authentication.

Table 8: Certificate-based authentication options

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`0`
ClientSSLCertificate (CSC)	No default value
ClientSSLKey (CSK)	No default value
HostNameInCertificate (HNIC)	No default value
Keystore (KS)	No default value
KeystorePassword (KSP)	No default value
Truststore (TS)	No default value
TruststorePassword (TSP)	No default value
ValidateServerCertificate (VSC)	`1` (Enabled)

Failover options

The following table summarizes the connection options that control how failover works with the driver.

Table 9: Failover options

Attribute (Short Name)	Default
AlternateServers (ASRV)	No default value
ConnectionRetryCount (CRC)	`0`
ConnectionRetryDelay (CRD)	`3`
FailoverGranularity (FG)	`0` (Non-Atomic)
FailoverMode (FM)	`0` (Connection)
FailoverPreconnect (FP)	`0` (Disabled)
LoadBalancing (LB)	`0` (Disabled)

Timeout options

The following table summarizes timeout options.

Table 10: Timeout options

Attribute (Short Name)	Default
LoginTimeout (LT)	`15`
QueryTimeout (QT)	`0`

Pooling options

The following table summarizes pooling options.

Table 11: Pooling options

Attribute (Short Name)	Default
ConnectionReset (CR)	`0` (Disabled)
LoadBalanceTimeout (LBT)	`0` (Disabled)
MaxPoolSize (MXPS)	`100`
MinPoolSize (MNPS)	`0`
Pooling (POOL)	`0` (Disabled)

Proxy server options

The following table summarizes proxy server options.

Table 12: Proxy server options

Attribute (Short Name)	Default
ProxyHost (PXHN)	No default value
ProxyMode (PXM)	0 (None)
ProxyPassword (PXPW)	No default value
ProxyPort (PXPT)	0
ProxyUser (PXU)	No default value

Additional options

The following table summarizes additional options.

Table 13: Additional options

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1
BatchMechanism (BM)	3 (Copy)
CallEscapeBehavior (CEB)	2
EnableKeysetCursors (EKC)	0 (Disabled)
EnableDescribeParam (EDP)	1 (Enabled)
ExtendedColumnMetaData (ECMD)	0 (Disabled)
ExtendedOptions (XO)	No default value
FetchRefCursors (FRC)	1 (Enabled)
FetchSize (FS)	50000
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
IANAAppCodePage (IACP) LINUX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	No default value
KeepAlive (KA)	0 (Disabled)
KeysetCursorOptions (KCO)	0 (RowID Columns)
MaxCharSize (MCS)	No default value

Attribute (Short Name)	Default
MaxLongVarcharSize (MLVS)	No default value
MaxVarcharSize (MVS)	No default value
ReportCodepageConversionErrors (RCCE)	`0` (Ignore Errors)
ShowSelectableTables (SST)	`1` (Enabled)
TransactionErrorBehavior (TEB)	`1` (Rollback Transaction)
UnboundedNumericPrecision (UNP)	`-1`
UnboundedNumericScale (UNS)	`-1`
UseDeclareFetch (UDF)	`0` (Disabled)
XMLDescribeType (XDT)	`-10`

For details, see the following topics:

- AllowedOpenSSLVersions
- Alternate Servers
- Application Using Threads
- Authentication Method
- Batch Mechanism
- Call Escape Behavior
- Client SSL Certificate
- Client SSL Key
- Connection Pooling
- Connection Reset
- Connection Retry Count
- Connection Retry Delay
- CryptoLibName
- Crypto Protocol Version
- Data Source Name
- Database Name
- Description
- Enable Keyset Cursors
- Enable SQLDescribeParam

- Encryption Method
- Extended Column MetaData
- Extended Options
- Failover Granularity
- Failover Mode
- Failover Preconnect
- Fetch Ref Cursors
- Fetch Size
- Fetch TSWTZ as Timestamp
- Fetch TWFS as Time
- GSS Client Library
- Host Name
- Host Name In Certificate
- IANAAppCodePage
- Initialization String
- Key Password
- Key Store
- Key Store Password
- Keyset Cursor Options
- Load Balance Timeout
- Load Balancing
- Login Timeout
- Max Char Size
- Max Long Varchar Size
- Max Pool Size
- Max Varchar Size
- Min Pool Size
- Password
- Port Number
- Proxy Host
- Proxy Mode
- Proxy Password
- Proxy Port

- Proxy User
- Query Timeout
- Report Codepage Conversion Errors
- Service Principal Name
- Show Selectable Tables
- SSLLibName
- TCP Keep Alive
- Transaction Error Behavior
- Trust Store
- Trust Store Password
- Unbounded Numeric Precision
- Unbounded Numeric Scale
- Use Declare Fetch
- User Name
- Validate Server Certificate
- XML Describe Type

AllowedOpenSSLVersions

Attribute

AllowedOpenSSLVersions (AOV)

Purpose

Important: Version 1.0.2 of the OpenSSL library has reached the end of its product life cycle and is no longer receiving security updates. Best security practices dictate that you use the latest version of the library.

Determines which version of the OpenSSL library file the driver uses for data encryption. Although the latest version of the OpenSSL library is the most secure, some characteristics of the library can cause connections to certain databases to fail. This option allows you to continue using older versions of the OpenSSL library while you transition your environment to support the latest version.

Valid Values

`latest|`*`openssl_version_number`*`[[,`*`openssl_version_number`*`]...]`

where:

`openssl_version_number`

is the version number for the OpenSSL library file to be loaded by the driver, for example, `1.0.2`. When more than one version is specified, the driver will first attempt to load the first version listed. If the driver is unable to locate and load this file, it will attempt to load the next version in the value. The driver currently supports versions `1.1.1` and `1.0.2`. Refer to the installed readme for latest supported versions.

Behavior

If set to `latest`, the driver loads the latest installed version of the OpenSSL library file provided by Progress.

If set to *`openssl_version_number`*, the driver loads the specified version of the OpenSSL library file. This value is used to specify a version other than the latest.

Notes

- This option is ignored if OpenSSL library files are specified using the CryptoLibName and SSLLibName options.
- This option works only with OpenSSL library files provided by Progress and user supplied OpenSSL library files that match Progress's naming convention and installation location.
- This option works only for installations using the default directory structure.
- Consult your database administrator concerning the security settings of your server.

Default

`1.1.1,1.0.2`

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:

```
AllowedOpenSSLVersions=1.0.2
```

See also

Advanced tab

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=PostgreSQLServer:
PortNumber=5431:Database=Pgredb1,
HostName=255.201.11.24:PortNumber=5432:Database=Pgredb2)
```

Notes

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

Default

No default value

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 73

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

Specifies the method the driver uses to authenticate the user to the server when a connection is established. If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Important: When Kerberos is enabled, if the database user name differs from the domain user name, you are required to pass the database user name via the User Name (LogonID) option in the datasource or connection string.

Valid Values

`0 | 4 | 17| 18`

Behavior

If set to `0` (No Encryption), the driver sends the user ID and password in clear text to the server for authentication.

If set to `4` (Kerberos Authentication), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

If set to `17` (SSL Verify CA), the driver uses certificate-based authentication, which verifies client SSL certificate to authenticate the client to the server.

If set to `18` (SSL Verify Full), the driver uses certificate-based authentication, which verifies both client SSL certificate and host name to authenticate the client to the server.

Default

`0` (No Encryption)

GUI Tab

Security tab

See also

Authentication on page 61

Batch Mechanism

Attribute

BatchMechanism (BM)

Purpose

Determines the mechanism that is used to execute batch operations.

Valid Values

`1` | `2` | `3` | `4`

Behavior

If set to `1` (SingleRowInsert), the driver executes an insert statement for each row contained in a parameter array. Specify this value if you are experiencing out-of-memory errors when performing batch inserts.

If set to `2` (MultiRowInsert), the driver attempts to execute a single insert statement for all the rows contained in a parameter array. If the size of the insert statement exceeds the available buffer memory of the driver, the driver executes multiple statements. Specify this value for substantial performance gains over `1` (SingleRowInsert) when performing batch inserts.

If set to `3` (Copy), the driver uses the PostgreSQL COPY command to insert rows into the target table. Specify this value for substantial performance gains over `1` (SingleRowInsert) when performing batch inserts.

If set to `4` (NativeBatch), the driver uses the PostgreSQL native batch protocol to insert all batched parameters.

Default

`3` (Copy)

Notes

- Batch Mechanism determines the mechanism used to perform batch inserts only. For update and delete batch operations, the driver uses the native batch mechanism to handle the request.
- When `BatchMechanism=3`, substantial performance gains can be made. However, the following limitations apply:
  - Individual update counts are not returned. However, the total number of inserted rows are returned the execution of a batch operation.
  - The entire batch insert is ATOMIC. If any issues are encountered, the entire operation fails and no rows are inserted.

GUI Tab

Advanced tab

See Also

See Performance considerations on page 73 for details.

Call Escape Behavior

Attribute

CallEscapeBehavior (CEB)

Purpose

Determines whether the driver calls a user-defined function or a stored procedure.

Valid Values

`0 | 1 | 2`

Behavior

If set to `0`, the driver calls a user-defined function.

If set to `1`, the driver calls a stored procedure.

If set to `2`, the driver determines whether to call a user-defined function or a stored procedure based on whether a return value parameter is specified. If a return value parameter is specified, the driver calls a user-defined function. If not, the driver calls a stored procedure.

Default

`2`

GUI Tab

Advanced tab

Client SSL Certificate

Attribute

ClientSSLCertificate (CSC)

Purpose

The full path of the client SSL certificate to be used for authenticating the client to the server when certificate-based authentication is enabled (`AuthenticationMethod=17` or `AuthenticationMethod=18`).

Valid Values

`string`

where:

`string`

  is the full path of the client SSL certificate.

Default

No default value

GUI Tab

Security tab

See also

Authentication on page 61

Client SSL Key

Attribute

ClientSSLKey (CSK)

Purpose

The key that is used to access the client SSL certificate when authenticating the client to the server when certificate-based authentication is enabled (`AuthenticationMethod=17` or `AuthenticationMethod=18`).

Valid Values

`key`

where:

`key`

is a valid key for the client SSL certificate.

Default

No default value

GUI Tab

Security tab

See also

Authentication on page 61

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 73

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 73

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

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_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

Notes

- 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:

```
C:\Program Files\Progress\DataDirect\ODBC_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

See Advanced tab for details.

See also

SSLLibName on page 131

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, behavior is determined by the setting of the EncryptionMethod connection option.

Valid Values

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

where:

`cryptographic_protocol`

is one of the following cryptographic protocols:

`TLSv1.2` | `TLSv1.1` | `TLSv1` | `SSLv3` | `SSLv2`

Caution: Good security practices recommend using TLSv1 or higher, due to known vulnerabilities in the SSLv2 and SSLv3 protocols.

Example

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

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

Notes

- This option is ignored if Encryption Method is set to 0 - No Encryption.
- Consult your database administrator concerning the data encryption settings of your server.

Default

`TLSv1.2,TLSv1.1,TLSv1`

GUI Tab

Security tab

See also

TLS/SSL encryption on page 65

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

No default value

GUI Tab

General tab

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

No default value

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

No default value

GUI Tab

General tab

Enable Keyset Cursors

Attribute

EnableKeysetCursors (EKC)

Purpose

Determines whether the driver emulates keyset cursors to provide scrollable keyset cursors to an ODBC application.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver emulates keyset cursors.

If set to `0` (Disabled), the driver does not emulate keyset cursors. If an application requests a keyset cursor and this option is set to 0, the driver uses a static cursor and returns a message that a different value was used.

Default

`0` (Disabled)

GUI Tab

Advanced tab

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), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

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

- For values `1` and `6`, the SSL protocol used is determined by the setting of the Crypto Protocol Version connection option.
- This connection option can affect performance.

Default

`0` (No Encryption)

GUI Tab

Security tab

See also

Crypto Protocol Version on page 100

Performance considerations on page 73

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

Extended Options

Attribute

ExtendedOptions (xo)

Purpose

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

Valid Values

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

where:

`option`

is a connection option.

`value`

is the value or setting of the connection option.

Default Value

No default value

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 Ref Cursors

Attribute

FetchRefCursors (FRC)

Purpose

Determines whether the driver returns refcursors from stored procedures as results sets.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver returns refcursors from stored procedures as result sets. The driver fetches all the data from the refcursor and then closes the refcursor. If a stored procedure returns multiple refcursors, the driver generates multiple result sets, one for each refcursor returned.

If set to `0` (Disabled), the driver returns the cursor name for refcursors. The application must fetch the actual data from the refcursor using the cursor name and must close the cursor before additional processing can be done on the statement. The application must close the cursor regardless of whether it actually fetches data from the cursor.

Default

`1` (Enabled)

GUI Tab

Advanced tab

Fetch Size

Attribute

FetchSize (FS)

Purpose

Specifies the maximum number of rows the driver returns in a network round trip when Use Declare Fetch connection option is set to 1 (`UseDeclareFetch=1`).

Valid Values

x

where

x

is a positive integer between 1 and 2,147,483,647 that indicates the maximum number of rows the driver returns in a network round trip. If no value or a value outside this range is specified, the driver sets Fetch Size to 50,000.

Notes

- `Fetch` and `A7` are the aliases for the Fetch Size connection option.
- Fetch Size can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.

Default

`50000`

GUI Tab

Advanced tab

Use Declare Fetch

See also

Performance considerations on page 73

Fetch TSWTZ as Timestamp

Attribute

FetchTSWTZasTimestamp (FTSWTZAT)

Purpose

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.

Valid Values

`0` | `1`

Behavior

If set to `1` (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 set to `0` (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

`0` (Disabled)

GUI Tab

Advanced tab

Fetch TWFS as Time

Attribute

FetchTWFSasTime (FTWFSAT)

Purpose

Determines whether the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME or SQL_TYPE_TIMESTAMP.

Valid Values

`0 | 1`

Behavior

If set to `1` (Enabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME. The fractional seconds portion of the value is truncated.

If set to `0` (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP. The fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp.

Notes

- When returning time with fractional seconds data as SQL_TYPE_TIMESTAMP, the Year, Month and Day parts of the timestamp must be set to zero.

Default

`0` (Disabled)

GUI Tab

Advanced tab

GSS Client Library

Attribute

GSSClient (GSSC)

Purpose

Specifies the name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

The driver uses the path defined by the PATH environment variable for loading the specified client library.

Valid Values

`native` | *`client_library`*

where:

`client_library`

is a GSS client library installed on the client.

Behavior

If set to *`client_library`*, the driver uses the specified GSS client library.

Note: For MIT Kerberos distributions, you must provide a full path to the MIT Library. For example, the 64-bit version for Windows would use the following value: `C:\Program Files\MIT\Kerberos\bin\gssapi64.dll`.

If set to `native`, the driver uses the GSS client for Windows Kerberos. All other users must provide the full path to the library name.

Default

`native`

GUI Tab

Security tab

Host Name

Attribute

HostName (HOST)

Purpose

The name or the IP address of the server to which you want to connect.

Valid Values

`server_name` | *`IP_address`*

where:

`server_name`

is the name of the server to which you want to connect.

`IP_address`

is the IP address of the server to which you want to connect.

The IP address can be specified in either IPv4 or IPv6 format. For details about these formats, see Using IP addresses on page 75.

Example

If your network supports named servers, you can specify a server name such as `MainServer`. Or, you can specify an IP address such as `199.226.224.34`.

Default

No default value

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 examines the subjectAltName values included in the certificate. If a dnsName value is present in the subjectAltName values, then the driver compares the value specified for Host Name In Certificate with the dnsName value. The connection succeeds if the values match. The connection fails if the Host Name In Certificate value does not match the dnsName value.

If no subjectAltName values exist or a dnsName value is not in the list of subjectAltName values, then the driver compares the value specified for Host Name In Certificate with the commonName part of the Subject name in the certificate. The commonName typically contains the host name of the machine for which the certificate was created. The connection succeeds if the values match. The connection fails if the Host Name In Certificate value does not match the commonName. If multiple commonName parts exist in the Subject name of the certificate, the connection succeeds if the Host Name In Certificate value matches any of the commonName parts.

If set to *#SERVERNAME#*, the driver compares the host server name specified as part of a data source or connection string to the dnsName or the commonName value.

Default

No default value

GUI Tab

Security tab

IANAAppCodePage

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 value you specify must match the database character encoding and the system locale.

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.

Notes

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

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

Advanced tab

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.

Example

To set the date format on every connection, specify:

```
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

No default value

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

No default value

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 (`EncryptionMethod=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

- The keystore and truststore files can be the same file.

Default

No default value

GUI Tab

Security tab

Key Store Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when SSL is enabled (`EncryptionMethod=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

No default value

GUI Tab

Security tab

Keyset Cursor Options

Attribute

KeysetCursorOptions (KCO)

Purpose

Determines which columns are used to comprise the keyset that the driver uses to create the initial keyset on which cursor operations are based. PostgreSQL does not offer a true row identifier column; the driver instead uses a hidden system column provided by the PostgreSQL database, ctid. Because the database might reassign the ID following a Vacuum operation, the driver can be configured to also include other columns to help ensure that data integrity is maintained.

Valid Values

`0 | 1`

Behavior

If set to `1` (RowID and Searchable Columns), the driver uses a combination of every non-LOB column in the Select list and the ctid ahidden column to build the keyset. By adding other Select list fields to the keyset, the driver is able to indicate the row cannot be found if the IDs change following a Vacuum operation.

If set to `0` (RowID Columns), the driver uses the ctid hidden system column.

Notes

- This option has no effect unless the EnableKeysetCursors (EKC) connection option is enabled.

Default

`0` (RowID) Columns

GUI Tab

Advanced 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 73

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 `10485760`

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, `10485760`.

Default

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

GUI Tab

Advanced tab

Max Long Varchar Size

Attribute

MaxLongVarcharSize (MLVS)

Purpose

Specifies the maximum size of columns of type SQL_LONGVARCHAR 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_LONGVARCHAR data type.

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 73

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 73

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

No default value

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

`5432`

GUI Tab

General tab

Proxy Host

Attribute

ProxyHost (PXHN)

Purpose

Specifies the host name and possibly the domain of the proxy server. The value specified can be a host name, a fully qualified domain name, or an IPv4 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 must be specified in IPv4 format.

Default

No default value

Notes

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

GUI Tab

General tab

See Also

- Proxy Mode on page 126
- Proxy Password on page 126
- Proxy Port on page 127
- Proxy User on page 128

Proxy Mode

Attribute

ProxyMode (PXM)

Purpose

Determines whether the driver connects to your data source endpoint through an HTTP proxy server.

Valid Values

`0` | `1`

Behavior

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

If set to `1` (HTTP), the driver connects to the data source endpoint through the HTTP proxy server specified by the Proxy Host connection option.

Default

`0` (NONE)

GUI Tab

General tab

See Also

- Proxy Host on page 125
- Host Name on page 112
- Proxy Password on page 126
- Proxy Port on page 127
- Proxy User on page 128

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 is configured to require authentication.

Default

No default value

GUI Tab

General tab

See Also

- Proxy Host on page 125
- Proxy Mode on page 126
- Proxy Port on page 127
- Proxy User on page 128

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 125
- Proxy Mode on page 126
- Proxy Password on page 126
- Proxy User on page 128

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

No default value

GUI Tab

General tab

See Also

- Proxy Host on page 125
- Proxy Mode on page 126
- Proxy Password on page 126
- Proxy Port on page 127

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

Service Principal Name

Attribute

ServicePrincipalName (SPN)

Purpose

Specifies the service principal name to be used by the driver for Kerberos authentication.

Valid Values

`servicePrincipalName`

where:

`servicePrincipalName`

is a valid service principal name.

If unspecified, the value of the Network Address option is used as the service principal name.

Notes

- If Authentication Method is set to `0`, the value of the Service Principal Name option is ignored.

Default

No default value

GUI Tab

Security 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 SSL library contains the implementations of TLS/SSL protocols the driver uses for data encryption.

This option allows you to designate a different 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_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

Notes

- 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_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

See Advanced tab for details.

See also

CryptoLibName on page 99

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 PostgreSQL server does not allow any operations on the connection except for rolling back the transaction.

Valid Values

`0 | 1 | 2`

Behavior

If set to `0` (None), 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 Transaction), 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.

If set to `2` (Rollback Savepoint), the driver rolls back the transaction to the last savepoint when an error is detected. In manual commit mode, the driver automatically sets a savepoint after each statement issued. This value makes transaction behavior resemble that of most other database system types, but uses more resources on the database server and may incur a slight performance penalty.

Default

`1` (Rollback Transaction)

GUI Tab

Advanced tab

Trust Store

Attribute

Truststore (TS)

Purpose

Specifies either the path and file name of the truststore file or the contents of the TLS/SSL certificates to be used when SSL is enabled (`Encryption Method=1`) and server authentication is used.

Valid Values

`*truststore_directory*\*filename*` | `data://-----BEGIN CERTIFICATE-----*certificate_content*-----END CERTIFICATE-----`

where:

`truststore_directory`

is the path to the directory where the truststore file is located.

`filename`

is the file name of the truststore file.

`certificate_content`

is the content of the TLS/SSL certificate.

Notes

- If you do not specify the path to the directory that contains the truststore file, the current directory is used for authentication.
- The keystore and truststore files may be the same file.
- When specifying content for multiple certificates, secify the content of each certificate between `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`. For example:

  ```
  -----BEGIN CERTIFICATE-----certificatecontent1-----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----certificatecontent2-----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----certificatecontent3-----END CERTIFICATE-----
  ```

  Note that the number of dashes (`-----`) must be the same before and after both `BEGIN CERTIFICATE` and `END CERTIFICATE`.
- When specifying the certificate content for authentication, do not specify the truststore password. Since the truststore file is not required to be stored on the disk when the certificate content is specified directly, the driver need not unlock its contents.
- The Trust Store field on the Driver setup dialog supports content up to 8192 characters in length. For specifying certificate content longer than 8192 characters, edit the registry and manually add the entry to the DSN.

Default

No default value

GUI Tab

Security tab

Trust Store 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

No default value

GUI Tab

Security tab

Unbounded Numeric Precision

Attribute

UnboundedNumericPrecision (UNP)

Purpose

Specifies the precision for unbounded NUMERIC columns when they are described within the column, parameter, result set, or table metadata. Executing aggregation operations (for example, sum or avg) on bounded NUMERIC columns often results in the server returning the aggregate column as an unbounded NUMERIC column. When this occurs, this option defines the precision for the aggregate column.

Valid Values

`precision`

where:

`precision`

is an integer from `-1` to `1000`.

Default

`-1`

GUI Tab

Advanced tab

Unbounded Numeric Scale

Attribute

UnboundedNumericScale (UNS)

Purpose

Specifies the scale for unbounded NUMERIC columns when they are described within the column, parameter, result set, or table metadata. Executing aggregation operations (for example, sum or avg) on bounded NUMERIC columns often results in the server returning the aggregate column as an unbounded NUMERIC column. When this occurs, this option defines the scale for the aggregate column.

Valid Values

`scale`

where:

`scale`

is an integer from `-1` to the value set for the Unbounded Numeric Precision connection option.

Notes

- The driver returns the scale specified in this option for the affected columns regardless of the number of decimal digits in a value. If a value contains fewer digits to the right of the decimal than the specified scale, the remaining digits are automatically returned as padded 0s. For example, if your scale is set to 6 and your value is 22.22, the value returned is 22.220000.

Default

`-1`

GUI Tab

Advanced tab

Use Declare Fetch

Attribute

UseDeclareFetch (UDF)

Purpose

Determines whether the driver attempts to return a result set in a single fetch or across multiple fetches. If multiple fetches are used, the size of the fetch is determined by the setting of the Fetch Size (FetchSize) option. For large result sets, retrieving rows in multiple fetches can improve response time and thereby prevent possible timeouts.

Valid Values

`0` | `1`

Behavior

If set to `0` (Disabled), the driver returns the complete result set in a single fetch.

If set to `1` (Enabled), the driver returns the result set in multipe fetches. The maximum number of rows to be returned in a fetch can be specified using the Fetch Size connection option.

Notes

- `B6` is an alias for the Use Declare Fetch connection option.
- The Use Declare Fetch server-side cursor is not supported with:
  - SELECT FOR UPDATE queries
  - SELECT INTO queries
  - Stored procedures
  - Batch queries (queries separated with semicolons)
  - Keyset cursors
- When Use Declared Fetch is enabled, the Numeric field values are returned in textual format.
- If you enable the Use Declare Fetch option (`UseDeclareFetch=1`) when connecting to a Yellowbrick server and attempt to execute a SQL query with aggregate functions, such as count() and groupby(), the driver will return the following error: "Unsupported node for serialization."

Default

0 (Disabled)

GUI Tab

Advanced tab

Fetch Size

See also

Performance considerations on page 73

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.

Important: When Kerberos is enabled, if the database user name differs from the domain user name, you are required to pass the database user name via the User Name (LogonID) option in the datasource 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

XML Describe Type

Attribute

XMLDescribeType (XDT)

Purpose

The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See Using the XML Data Type on page 31 for further information about the XML data type.

Valid Values

`-4` | `-10`

Behavior

If set to `-4` (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to `-10` (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

`-10`

GUI Tab

Advanced tab