Progress DataDirect for ODBC for Db2 Wire Protocol Driver

User's Guide

Release 8.0.2

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Db2 Wire Protocol Driver

The Progress® DataDirect® for ODBC™ for Db2™ Wire Protocol driver (the Db2 Wire Protocol driver) supports the following Db2 database servers:

- Db2 for i
- Db2 for Linux, UNIX, and Windows
- Db2 for z/OS
- Db2 Hosted
- Db2 Warehouse on Cloud (formerly dashDB)

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?

Support and Certifications

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

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

Changes for 8.0.2 GA

- **Driver Enhancements**
  - The default version of the OpenSSL library has been upgraded to 3.5.6. As part of this upgrade, earlier version of the OpenSSL 3.0 library continues to be supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files:
    - Windows: `ivopenssl.dll` and `ddopenssl.dll`
    - Unix: `ivopenssl.so` and `ddopenssl.so`
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The driver has been enhanced to support the TLSv1.3 cryptographic protocol. As part of this enhancement, the default cryptographic protocols enabled by the driver have been updated to TLSv1.3 and TLSv1.2. See Crypto Protocol Version on page 131 for more information.
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

    The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL server authentication on page 73 and TLS/SSL client authentication on page 78 for details.
  - The driver has been enhanced to support the Windows certificate store for TLS/SSL server authentication. See TLS/SSL server authentication on page 73 for details.
  - The driver has been enhanced to support the BINARY and VARBINARY data types.
  - 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 165 and Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 76 for details.

- The driver has been enhanced to support Select queries with parameterized arrays.
- The driver has 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.
- A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and odbc.ini files. At connection, the driver decrypts these passwords and passes them to the data source as required. See Password Encryption Tool (UNIX/Linux only) on page 67 for details.

- **Changed Behavior**
  - The TLSv1.1 and TLSv1.0 cryptographic protocols are now disabled by default and have been removed as selectable values for the Crypto Protocol Version (CryptoProtocolVersion) option on the Setup dialog. These protocols are no longer considered secure and, therefore, are no longer recommended for use. However, the driver still supports TLSv1.1 and TLSv1.0 for legacy servers that do not support more secure protocols. See Crypto Protocol Version on page 131 for more information.
  - The Allowed OpenSSL Versions (AllowedOpenSSLVersions) connection option has been deprecated.
  - The product no longer includes version 1.1.1 of the OpenSSL library. The library will reach the end of its product life cycle in September 2023 and will not receive any security updates after that. Note that continuing to use the library after September 2023 can potentially expose you to security vulnerabilities.

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

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

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

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

- 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.
- You must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

AIX requirements for 32-bit and 64-bit drivers

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

HP-UX requirements for 32-bit drivers

- The following processors are supported:
  - PA-RISC
  - Intel Itanium II (IPF)
- For PA-RISC: An application compatible with components that were built using HP aC++ 3.30 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads).
- For IPF: An application compatible with components that were built using HP aC++ 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads).

HP-UX requirements for 64-bit drivers

- Intel Itanium II (IPF) processor
- HP aC++ v. 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads).

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**:

- **For Db2 for z/OS and iSeries**:
  - **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`.
  - **IP Address**: Type the host name or the IP address of the machine where catalog tables are stored.
  - **TCP Port**: Type the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.
  - **Location Name**: Type the name of the Db2 location that you want to access.
  - **Collection**: Type the name of the current collection or library.
- **For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud**:
  - **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`.
  - **IP Address**: Type the host name or the IP address of the machine where catalog tables are stored.
  - **TCP Port**: Type the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.
  - **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 69.

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 18 provides connection string examples that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
   - Connection option descriptions on page 109 provides a complete list of supported options by functionality.
   - Performance considerations on page 95 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 Db2 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.
    - 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 60

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.

     For Db2 for z/OS and iSeries:

     ```
     [ODBC Data Sources]
     Db2=DataDirect 8.0 Db2 Wire Protocol

     [Db2]
     Driver=ODBCHOME/lib/xxdb228.yy
     ...
     Collection=payroll
     ...
     IPAddress=localhost
     ...
     Location=NYC
     ...
     LogonID=jsmith
     ...
     Password=secret
     ...
     TCPPort=50000
     ```

     For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud:

     ```
     [ODBC Data Sources]
     Db2=DataDirect 8.0 Db2 Wire Protocol

     [Db2]
     Driver=ODBCHOME/lib/xxdb228.yy
     ...
     Database=db2database
     ...
     IPAddress=localhost
     ...
     LogonID=jsmith
     ...
     Password=secret
     ...
     TCPPort=50000
     ```

     See Configuration through the system information (odbc.ini) file on page 61 for more information.

     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.

   - **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 60, DSN-less connections, for more information. For examples, see Connection string examples on page 18.

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

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

- Connection string examples on page 18 provides connection string examples that can be used to configure common functionality and features. You can modify these examples to create a string that best suites your environment.

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

- Connection option descriptions on page 109 provides a complete list of supported options by functionality.
- Performance considerations on page 95 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 60

User ID/password authentication

The following strings include the options used to connect using user ID/password (no encryption) authentication.

Important: To use user ID/ password authentication with encryption, set the Authentication Method option to one of the following values: 1 (Encrypt Password), 2 (Encrypt UID and Password), 7 (Encrypted Password AES), and 8 (Encrypted UID and Password AES). See "Authentication Method" for details.

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

For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=ip_address;TCPPort=tcp_port;
Database=database_name;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

For Db2 for z/OS and iSeries:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=ip_address;TCPPort=tcp_port;
Location=location_name;Collection=collection_name;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

where:

ip_address

specifies the host name or the IP address of the machine where catalog tables are stored.

tcp_port

specifies the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.

database_name

specifies the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.

location_name

specifies the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.

collection_name

specifies the current collection or library. Valid only on Db2 for z/OS and Db2 for i.

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 connecting to Db2 for Linux, UNIX, and Windows using user ID/password (no encryption) authentication.

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=localhost;TCPPort=50000;
Database=db2data;LogonID=jsmith;Password=secret;
```

See also

Authentication Method on page 121
Using a connection string on page 60
Connection option descriptions on page 109

Kerberos authentication

The following strings include the options used to connect using the Kerberos authentication.

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

For Db2 for Linux, UNIX, and Windows:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;AuthenticationMethod=4;
IPAddress=ip_address;TCPPort=tcp_port;Database=database_name;
GSSClient=gss_client_library;[attribute=value[;...]];
```

For Db2 for z/OS:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;AuthenticationMethod=4;
IPAddress=ip_address;TCPPort=tcp_port;Location=location_name;
Collection=collection_name;GSSClient=gss_client_library;
[attribute=value[;...]];
```

where:

ip_address

specifies the host name or the IP address of the machine where catalog tables are stored.

tcp_port

specifies the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default value is `50000`.

database_name

specifies the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows.

collection_name

specifies the current collection or library. Valid only on Db2 for z/OS.

location_name

specifies the name of the Db2 location that you want to access. Valid only on Db2 for z/OS.

gss_client_library

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

attribute=value

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

The following example connection string includes the options required for connecting to Db2 for Linux, UNIX, and Windows using the Kerberos authentication.

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;AuthenticationMethod=4;
IPAddress=123.456.78.90;TCPPort=50000;Database=Payroll;GSSClient=gss123;
```

See also

Using a connection string on page 60
Connection option descriptions on page 109

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.

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

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;
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 when connecting to Db2 for Linux, UNIX, and Windows.

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;
AlternateServers=(IPAddress=localhost:TCPPort=50000:Database=db2data,
IPAddress=123.456.78.90:TCPPort=50001:Database=db2data1);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0;
LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 60
Connection option descriptions on page 109
Failover support on page 83
Load Balancing on page 151
Failover Mode on page 141

TLS/SSL client authentication

The following strings configure 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, the driver uses user ID/password authentication.

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

For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=ip_address;HostNameInCertificate=host_name_in_certificate;
TCPPort=tcp_port;Database=database_name;
Keystore=keystore_name;KeystorePassword=keystore_password;
ValidateServerCertificate=validate_server_certificate;
EnableFIPS=enable_fips;LogonID=user_name;Password=password;
```

For Db2 for z/OS and iSeries:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=ip_address;HostNameInCertificate=host_name_in_certificate;
TCPPort=tcp_port;Location=location_name;Collection=collection_name;
Keystore=keystore_name;KeystorePassword=keystore_password;
ValidateServerCertificate=validate_server_certificate;
EnableFIPS=enable_fips;LogonID=user_name;Password=password;
```

where:

ip_address

specifies the host name or the IP address of the machine where catalog tables are stored.

host_name_in_certificate

specifies a host name for certificate validation.

tcp_port

specifies the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.

database_name

specifies the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.

collection_name

specifies the current collection or library. Valid only on Db2 for z/OS and Db2 for i.

location_name

specifies the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.

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. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.

enable_fips

determines whether the driver loads the FIPS provider or the default provider. When Enable FIPS is set to `1`, the driver loads the FIPS provider and when it is set to `0`, the driver loads the default provider. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2.

Note:

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
- Do not set the Truststore Password connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms" for more information.

---

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 required for connecting to Db2 for Linux, UNIX, and Windows using TLS/SSL client authentication.

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=localhost;TCPPort=50000;HostNameInCertificate=MySubjectAltName;
Database=db2data;Keystore=KeyStoreName;KeystorePassword=YourKSPassword;
ValidateServerCertificate=1;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 60
Connection option descriptions on page 109
TLS/SSL encryption on page 72
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 80

TLS/SSL server authentication

The following strings configure 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, the driver uses user ID/password authentication.

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

For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=ip_address;HostNameInCertificate=host_name_in_certificate;
TCPPort=tcp_port;Database=database_name;Truststore=truststore;
TruststorePassword=truststore_password;
ValidateServerCertificate=validate_server_certificate;
EnableFIPS=enable_fips;LogonID=user_name;Password=password;
```

For Db2 for z/OS and iSeries:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=ip_address;HostNameInCertificate=host_name_in_certificate;
TCPPort=tcp_port;Location=location_name;Collection=collection_name;
Truststore=truststore;TruststorePassword=truststore_password;
ValidateServerCertificate=validate_server_certificate;
EnableFIPS=enable_fips;LogonID=user_name;Password=password;
```

where:

ip_address

specifies the host name or the IP address of the machine where catalog tables are stored.

host_name_in_certificate

specifies a host name for certificate validation.

tcp_port

specifies the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.

database_name

specifies the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.

collection_name

specifies the current collection or library. Valid only on Db2 for z/OS and Db2 for i.

location_name

specifies the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.

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. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.

enable_fips

determines whether the driver loads the FIPS provider or the default provider. When Enable FIPS is set to `1`, the driver loads the FIPS provider and when it is set to `0`, the driver loads the default provider. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2.

Note:

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
- Do not set the Truststore Password connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms" for more information.

---

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 required for connecting to Db2 for Linux, UNIX, and Windows using TLS/SSL server authentication.

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EncryptionMethod=1;
IPAddress=localhost;TCPPort=50000;HostNameInCertificate=MySubjectAltName;
Database=db2data;Truststore=TrustStoreName;TruststorePassword=TSXYZZY;
ValidateServerCertificate=1;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 60
Connection option descriptions on page 109
TLS/SSL encryption on page 72
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 80

Version string information

The driver has a version string of the format:

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

or

XX.YY.ZZZZ(b*AAAA*, u*BBBB*)

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

XX.YY.ZZZZ(U*BBBB*)

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

XX.YY.ZZZZ

where:

XX is the major version of the product.

YY is the minor version of the product.

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

AAAA is the build number of the driver's 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 ivdb228.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. The HP-UX version of the tool, however, requires the full path.

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 Db2 data types map to the standard ODBC data types. Unicode support in Driver specifications on page 30 lists Db2 to Unicode data type mappings.

Table 1: Db2 Data Types

Db2	ODBC
Bigint[1]	SQL_BIGINT
Binary	SQL_BINARY
Blob[2]	SQL_LONGVARBINARY
Char	SQL_CHAR
Char() for Bit Data	SQL_BINARY
Clob[3]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decfloat[4]	SQL_DOUBLE
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY
Numeric	SQL_NUMERIC
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME

[1] Supported on Db2 V8.x and higher for Linux/UNIX/Windows, Db2 V9 and higher for Db2 for z/OS, and Db2 V5R3 and higher for Db2 for i.

[2] Supported on Db2 V8.x and higher for Linux/UNIX/Windows; Db2 for z/OS; and Db2 V5R3 and higher for Db2 for i.

[3] On Db2 for Linux/UNIX/Windows versions previous to V8.1 and Db2 V5R2 for Db2 for i, only the first 32 KB of the Clob data are returned when fetching, and only 32 KB can be inserted and updated.

[4] Supported only on Db2 V9 and higher for Linux/UNIX/Windows, Db2 V9 and higher for Db2 for z/OS, and Db2 i V6R1 for Db2 for i.

Db2	ODBC
Timestamp[5]	SQL_TYPE_TIMESTAMP
Timestamp With Time Zone[2, 6]	SQL_VARCHAR
Varbinary	SQL_VARBINARY
Varchar	SQL_VARCHAR
Varchar() for Bit Data	SQL_VARBINARY
XML[7]	SQL_LONGVARCHAR

Using the XML Data Type

By default, Db2 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 attribute XMLDescribeType (XDT) to SQL_LONGVARBINARY (-10) and bind the data as SQL_C_BINARY.

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:

  - SQLProcedures
  - SQLProcedureColumns

  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 Db2 Wire Protocol driver supports Unicode data types if the database was created with a multi-byte character set.

  The driver maps the Db2 data types to Unicode data types as shown in the following table:

Db2 Data Type	Mapped to. . .
Dbclob[8]	SQL_WLONGVARCHAR

[5] Timestamp values with a fractional seconds precision greater than 9 are described as the ODBC SQL_VARCHAR data type.
[6] Timestamp with Time Zone mapping changes based on the setting of the FetchTSWTZasTimestamp option only on Db2 V10 and higher for Db2 for z/OS.
[7] Supported only on Db2 V10 for z/OS.
[8] Supported on Db2 V8.x and higher for Linux/UNIX/Windows, Db2 V9 and higher for Db2 for z/OS, and Db2 V5R3 and higher for Db2 for i.

Db2 Data Type	Mapped to. . .
Graphic	SQL_WCHAR
Long Vargraphic	SQL_WLONGVARCHAR
Vargraphic	SQL_WVARCHAR

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

- **Isolation and lock levels**: The driver supports isolation level 0 (read uncommitted), isolation level 1 (read committed), isolation level 2 (repeatable read), isolation level 3 (serializable), and, on Db2 for i only, isolation level 4 (none).

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

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

Additional information

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

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

Troubleshooting

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

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

Contacting Technical Support

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

https://www.progress.com/support

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

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

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.
- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

Jan 2022, Release 8.0.2 for the Progress DataDirect for ODBC for Db2 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)

The Example application

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

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

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

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

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

Note:

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

To use the Example application:

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

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

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

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

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 DB2.tdc`

2. Copy the Tableau data source file into the following directory:

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

3. Open Tableau. If the **Connect** menu does not open by default, select **Data > New Data Source** or the Add New Data Source button  to open the menu.
4. From the **Connect** menu, select **Other Databases (ODBC)**.
5. The **Other Databases (ODBC)** 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 **Sign in** on the Other Databases (ODBC) 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 schema you want to use. The tables stored in this schema are now available for selection in the Table field.

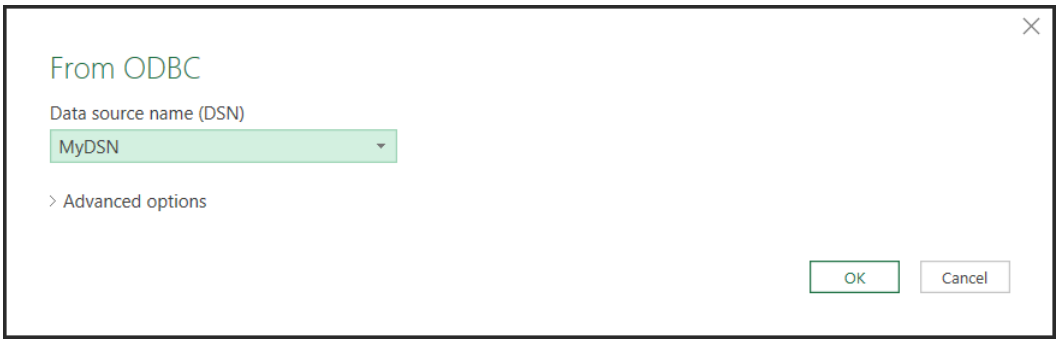
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

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.

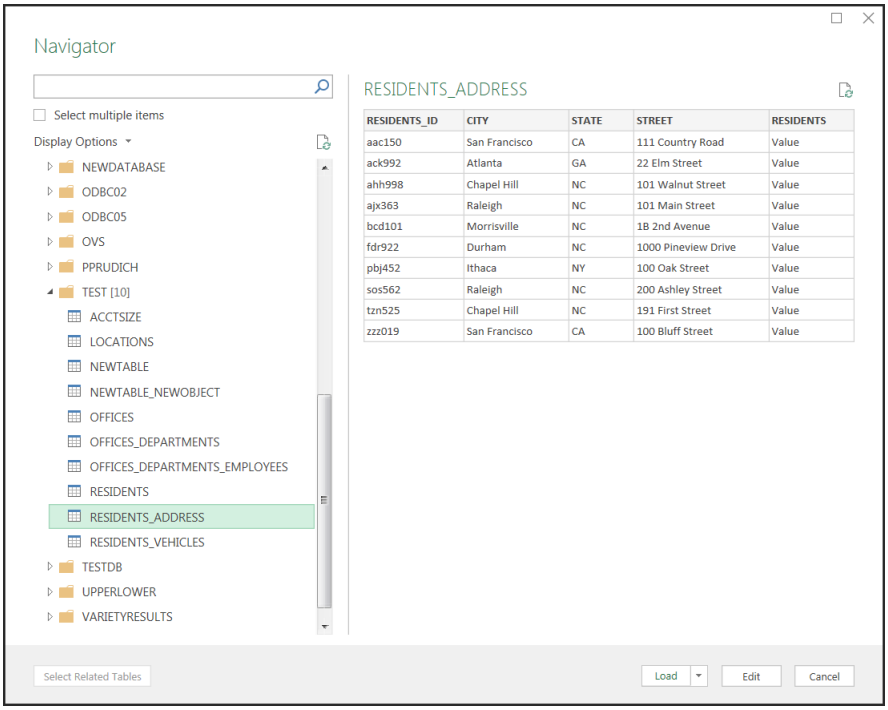


Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

4. You are prompted for logon credentials for your data source:

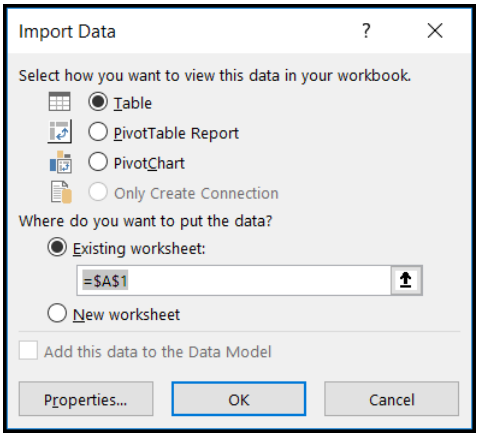
   - If your data source does not require logon credentials or if you prefer to specify your credentials using a connection string, select **Default or Custom** from the menu on the left. Optionally, specify your credential-related options using a connection string in the provided field. Click **Connect** 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.



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

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



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

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

3

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. 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
- Password Encryption Tool (UNIX/Linux only)
- Using a logon dialog box
- Authentication
- TLS/SSL encryption

- Failover support
- DataDirect Bulk Load
- 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 HP-UX IPF, 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/ivdb2xx.so
```

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

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

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 Db2 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:
   - **For Db2 for z/OS and iSeries**:
     - **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`.
     - **IP Address**: Type the host name or the IP address of the machine where catalog tables are stored.
     - **TCP Port**: Type the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.
     - **Location Name**: Type the name of the Db2 location that you want to access.
     - **Collection**: Type the name of the current collection or library.
   - **For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud**:
     - **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`.
     - **IP Address**: Type the host name or the IP address of the machine where catalog tables are stored.
     - **TCP Port**: Type the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.
     - **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 on page 46, provide values for options to configure advanced behavior.
6. If applicable, on the Security tab on page 48, configure security settings.
7. If applicable, on the Failover tab on page 51, configure failover data source settings.
8. If applicable, on the Pooling tab on page 53, 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.

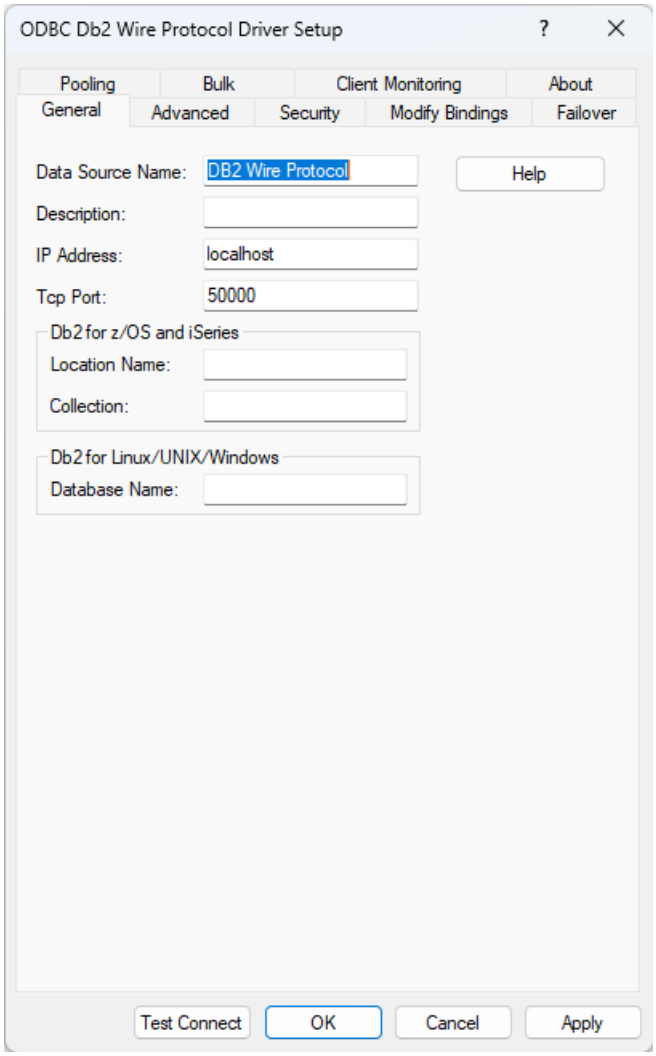
See also

Using a logon dialog box on page 68

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



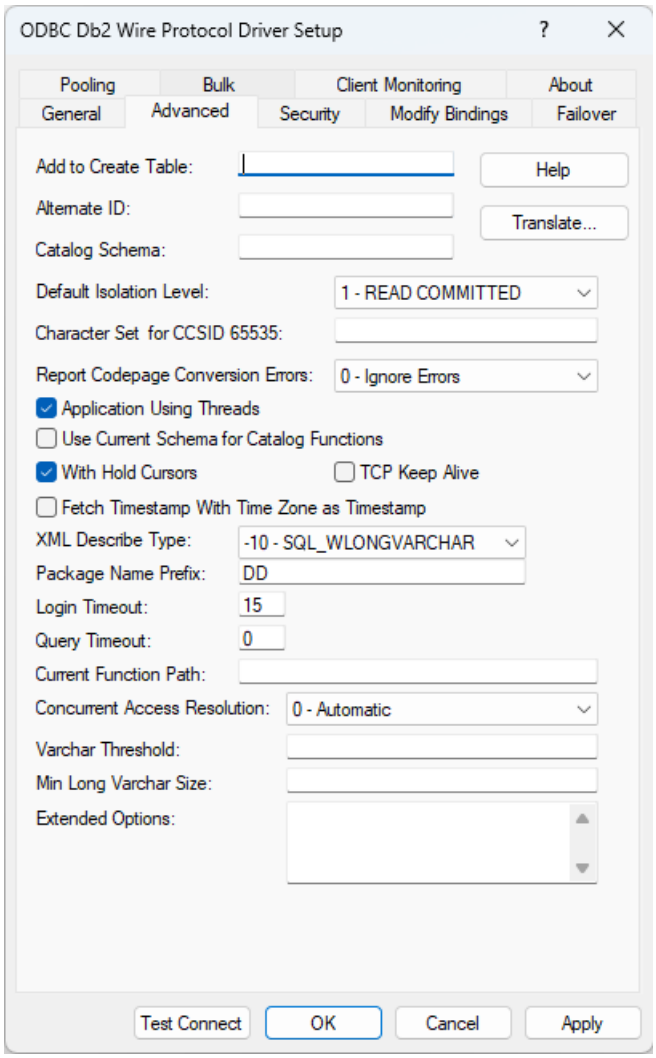
Connection Options: General	Default
Data Source Name on page 134	No default value
Description on page 137	No default value
IP Address on page 148	No default value
TCP Port on page 164	`50000`
Location Name on page 152	No default value

Connection Options: General	Default
Collection on page 127	No default value
Database Name on page 135	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



Connection Options: Advanced	Default
Add to Create Table on page 117	No default value
Alternate ID on page 118	No default value
Catalog Schema on page 124	No default value

Connection Options: Advanced	Default
Default Isolation Level on page 135	**1 - READ_COMMITTED**
Character Set for CCSID 65535 on page 124	`0`
Report Codepage Conversion Errors on page 162	**0 - Ignore Errors**
Application Using Threads on page 120	Enabled
Use Current Schema for Catalog Functions on page 167	Disabled
With Hold Cursors on page 170	Enabled
TCP Keep Alive on page 164	Disabled
Fetch Time Stamp With Time Zone as Timestamp on page 143	Disabled
XML Describe Type on page 170	**-10 - SQL_WLONGVARCHAR**
Package Name Prefix on page 157	`DD`
Login Timeout on page 152	`15`
Query Timeout on page 160	`0`
Current Function Path on page 133	No default value
Concurrent Access Resolution on page 128	**0 - Automatic**
Varchar Threshold on page 169	No default value
Min Long Varchar Size on page 154	No default value
Extended Options on page 140	No default value

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

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

See also

Configuring the driver using the GUI on page 43

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

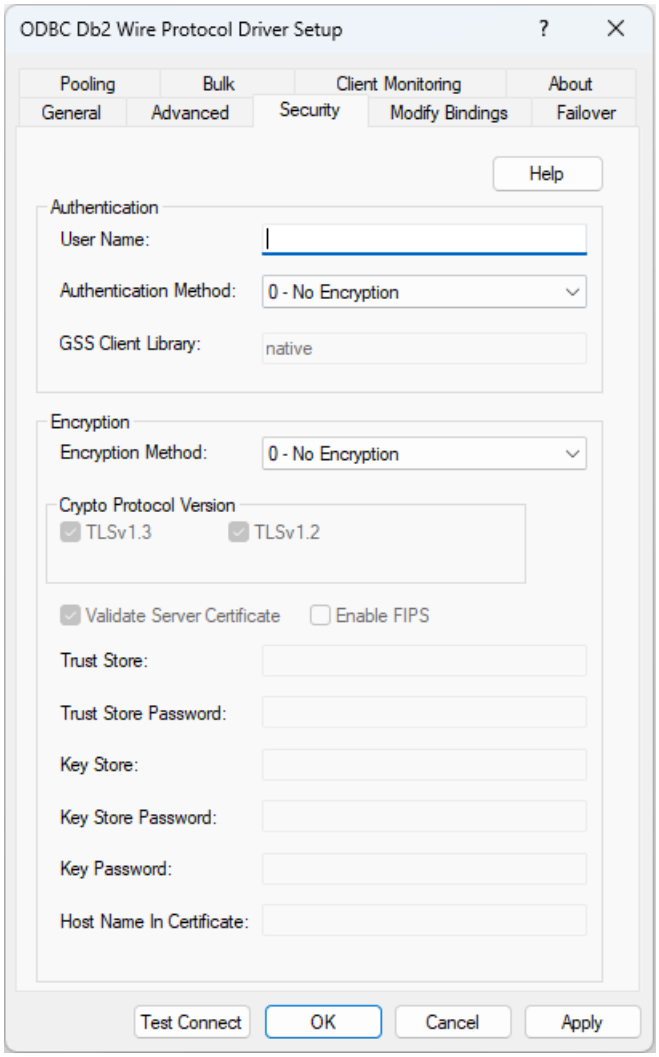


Table 2: Security Tab Connection Options

Connection Options: Security	Default
User Name on page 167	No default value
Authentication Method on page 121	**0-No Encryption**
GSS Client Library on page 145	`native`
Encryption Method on page 139	**0-No Encryption**
Crypto Protocol Version on page 131	`TLSv1.3,TLSv1.2`

Connection Options: Security	Default
Validate Server Certificate on page 168	Enabled
Enable FIPS on page 138	Disabled
Trust Store on page 165	No default value
Trust Store Password on page 166	No default value
Keystore on page 149	No default value
Key Store Password on page 150	No default value
Key Password on page 148	No default value
Host Name In Certificate on page 146	No default value

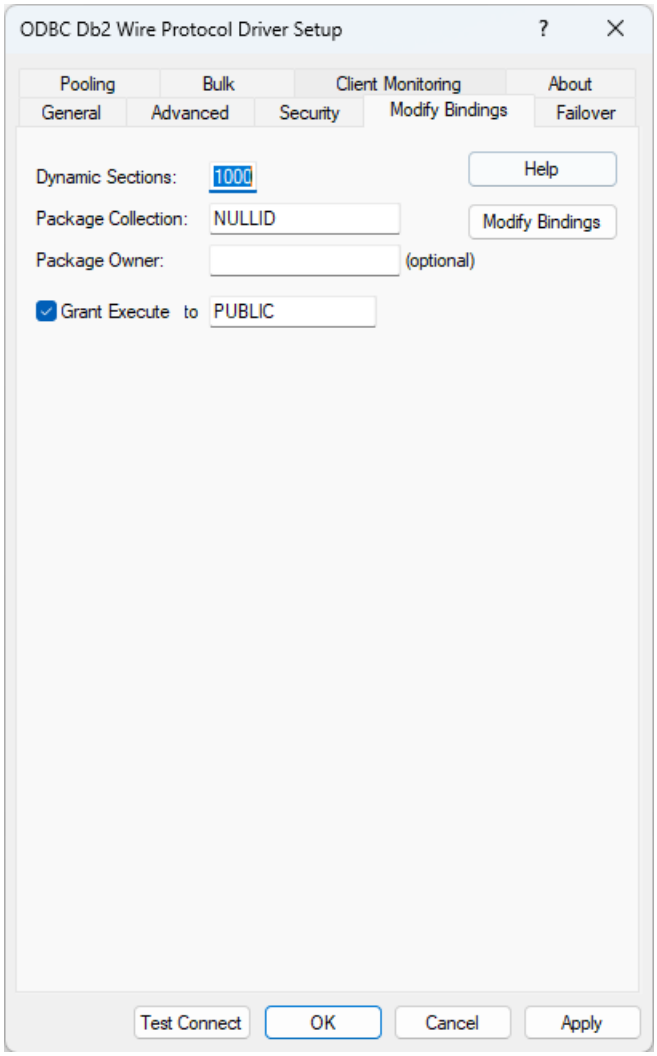
See also

TLS/SSL encryption on page 72

Modify Bindings tab

The Modify Bindings tab allows you to configure options for creating or modifying bind packages. The fields are optional unless otherwise noted. On this tab, provide values for the options in the following table; then, click **Apply**. Optionally, click **Modify Bindings** to create a bind package before connection or modify an existing one. If you do not click **Modify Bindings**, the driver uses the provided settings the next time it establishes a connection.

Figure 4: Modify Bindings tab



The Modify Bindings tab allows you to create or modify bind packages on the server accessed by the driver. If you connect with the driver before explicitly creating bind packages, the driver creates packages on the server automatically using the current values from the Setup dialog box. Alternatively, you can create a bind package before testing the connection. You can also modify bind packages after their creation from the Modify Bindings tab. You must also provide appropriate values for the options on the General tab.

Note: If you change any of the values on this tab after having initially created bind packages, you must rebind the packages. The changes are reflected only on new connections after rebinding.

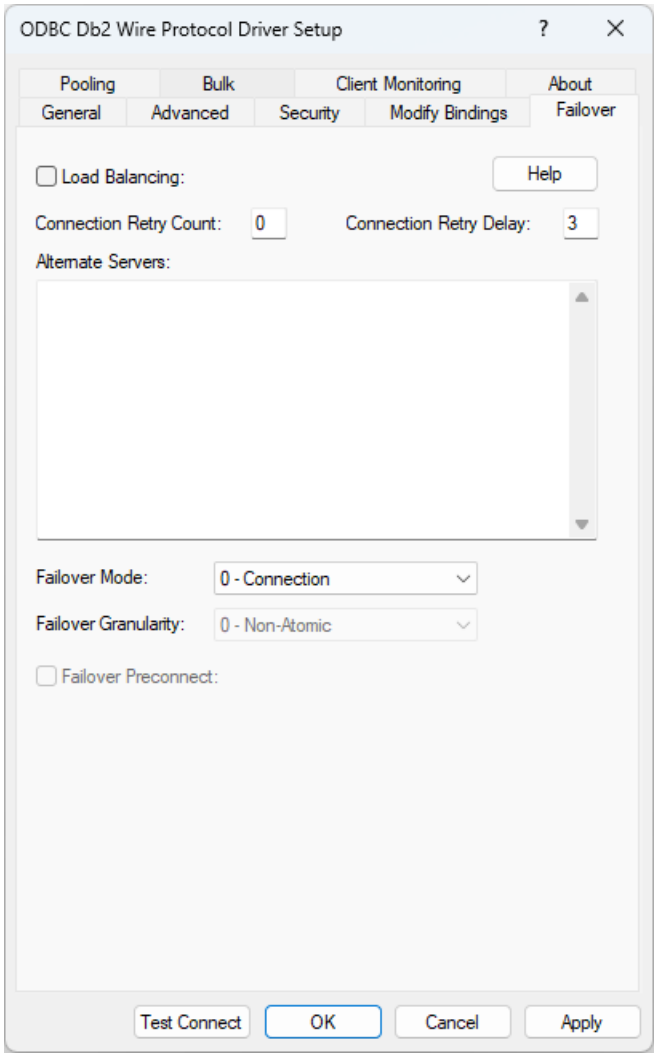
Table 3: Modify Bindings Tab Connection Options

Connection Options: Modify Bindings	Default
Dynamic Sections on page 137	`1000`
Package Collection on page 157	`NULLID`
Package Owner on page 158	No default value
Grant Execute to [check box] on page 144	Enabled
Grant Execute to [field] on page 144	`PUBLIC`

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 5: Failover tab



Connection Options: Failover	Default
Load Balancing on page 151	Disabled
Connection Retry Count on page 130	0
Connection Retry Delay on page 131	3
Alternate Servers on page 118	No default value
Failover Mode on page 141	**0 - Connection**
Failover Granularity on page 140	**0 - Non-Atomic**
Failover Preconnect on page 142	Disabled

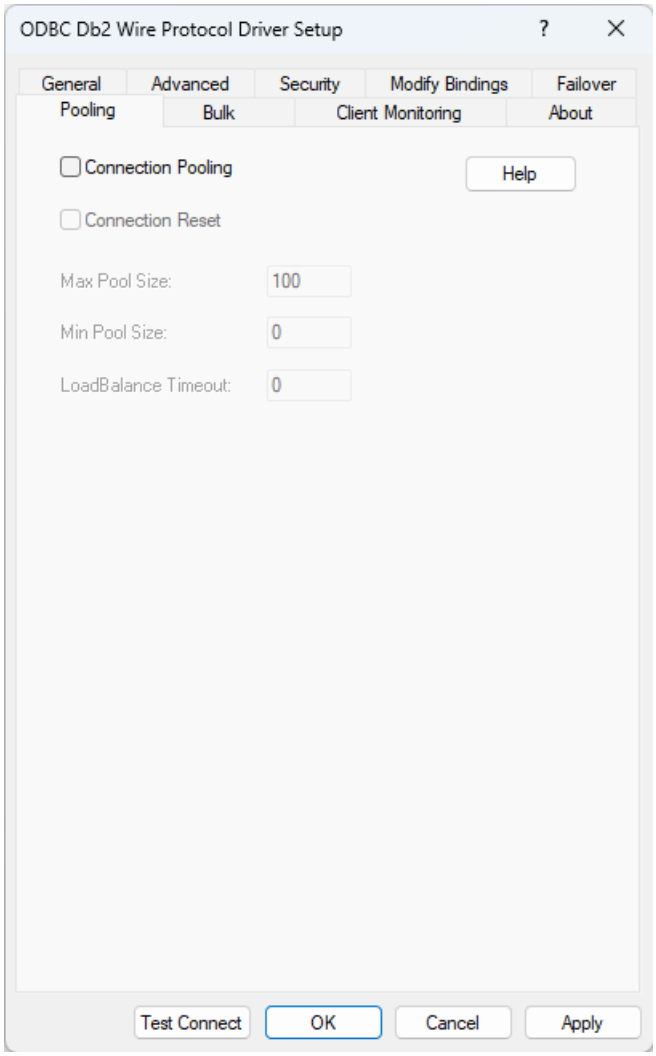
See also

Configuring the driver using the GUI on page 43

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 6: Pooling tab



Connection Options: Pooling	Default
Connection Pooling on page 128	Disabled
Connection Reset on page 129	Disabled
Max Pool Size on page 153	`100`
Min Pool Size on page 155	`0`
Load Balance Timeout on page 150	`0`

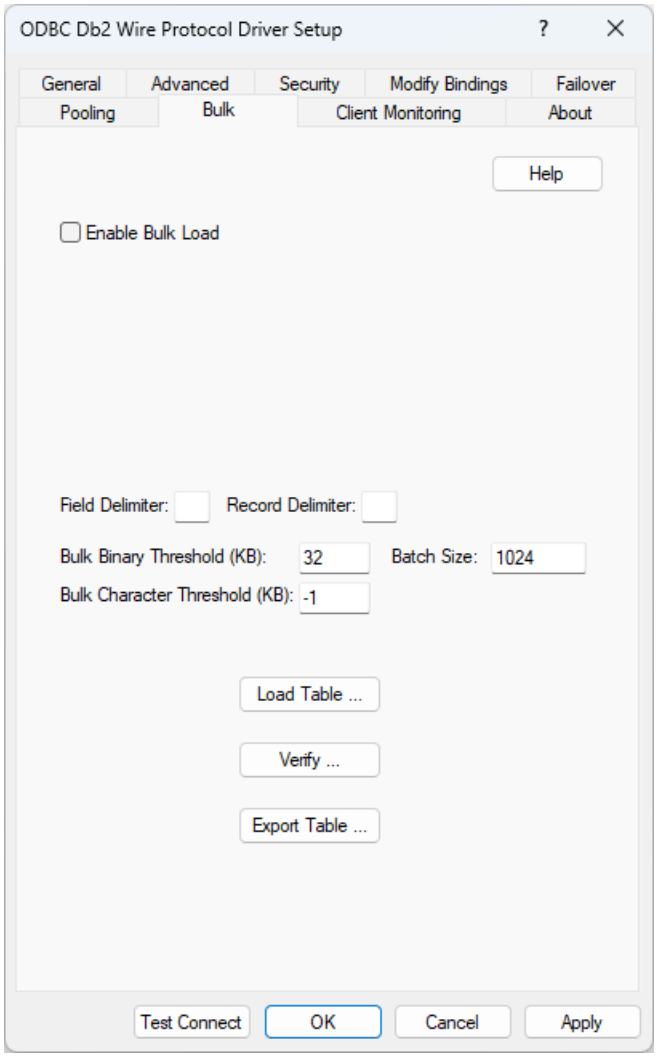
See also

Configuring the driver using the GUI on page 43

Bulk tab

The Bulk tab allows you to specify DataDirect Bulk Load 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 7: Bulk tab



Connection Options: Bulk	Default
Enable Bulk Load on page 138	Disabled
Field Delimiter on page 143	No default value
Record Delimiter on page 161	No default value
Bulk Binary Threshold on page 122	32

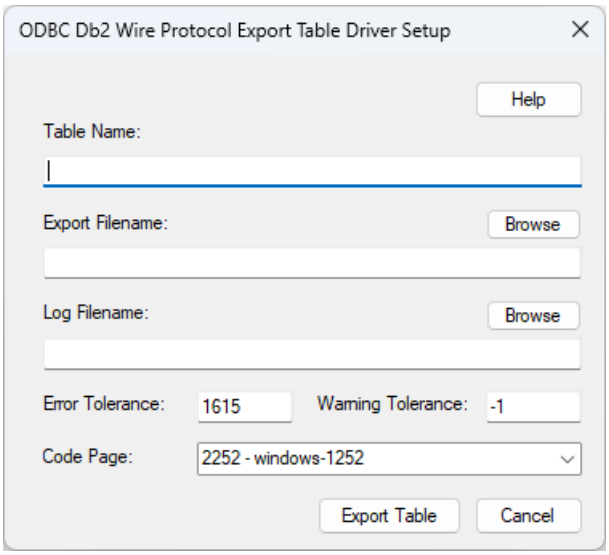
Connection Options: Bulk	Default
Batch Size on page 122	1024
Bulk Character Threshold on page 123	-1

If your application is already coded to use parameter array batch functionality, you can leverage DataDirect Bulk Load features through the Enable Bulk Load connection option. Enabling this option automatically converts the parameter array batch operation to use the database bulk load protocol.

If you are not using parameter array batch functionality, you can export data to a bulk load data file, verify the metadata of the bulk load configuration file against the structure of the target table, and bulk load data to a table. Use the following steps to accomplish these tasks.

1. To export data from a table to a bulk load data file, click **Export Table** from the Bulk tab. The Export Table dialog box appears.

   Figure 8: Export Table dialog box



   Both a bulk data file and a bulk configuration file are produced by exporting a table. The configuration file has the same name as the data file, but with an XML extension. See DataDirect Bulk Load on page 85 for details about these files.

   The bulk export operation can create a log file and can also export to external files. See External overflow files on page 93 for more information. The export operation can be configured such that if any errors or warnings occur:

   - The operation always completes.
   - The operation always terminates.
   - The operation terminates after a certain threshold of warnings or errors is exceeded.

   The following fields are available on the Export Table dialog box:

   - **Table Name**: A string that specifies the name of the source database table containing the data to be exported.
   - **Export Filename**: A string that specifies the path (relative or absolute) and file of the bulk load data file to which the data is to be exported. It also specifies the file name of the bulk configuration file. The file name must be the fully qualified path to the bulk data file. These files must not already exist; if one or both of them already exists, an error is returned.

- **Log Filename**: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. The file name must be the fully qualified path to the log file. Events logged to this file are:
  - Total number of rows fetched
  - A message for each row that failed to export
  - Total number of rows that failed to export
  - Total number of rows successfully exported

  Information about the load is written to this file, preceded by a header. Information about the next load is appended to the end of the file.

  If you do not specify a value for Log Filename, no log file is created.

- **Error Tolerance**: A value that specifies the number of errors to tolerate before an operation terminates. A value of `0` indicates that no errors are tolerated; the operation fails when the first error is encountered.

  The default of `-1` means that an infinite number of errors is tolerated.

- **Warning Tolerance**: A value that specifies the number of warnings to tolerate before an operation terminates. A value of `0` indicates that no warnings are tolerated; the operation fails when the first warning is encountered.

  The default of `-1` means that an infinite number of warnings is tolerated.

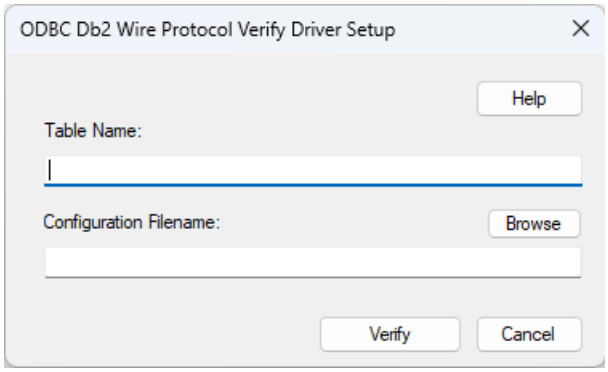
- **Code Page**: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See Character set conversions on page 93 for more information.

  The default value on Windows is the current code page of the machine. On UNIX/Linux, the default value is 4 (ISO 8559-1 Latin-1).

- Click **Export Table** to connect to the database and export data to the bulk data file or click **Cancel**.

  To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify**from the Bulk tab. See Verification of the bulk load configuration file on page 91 for details. The Verify dialog box appears.

  Figure 9: Verify dialog box

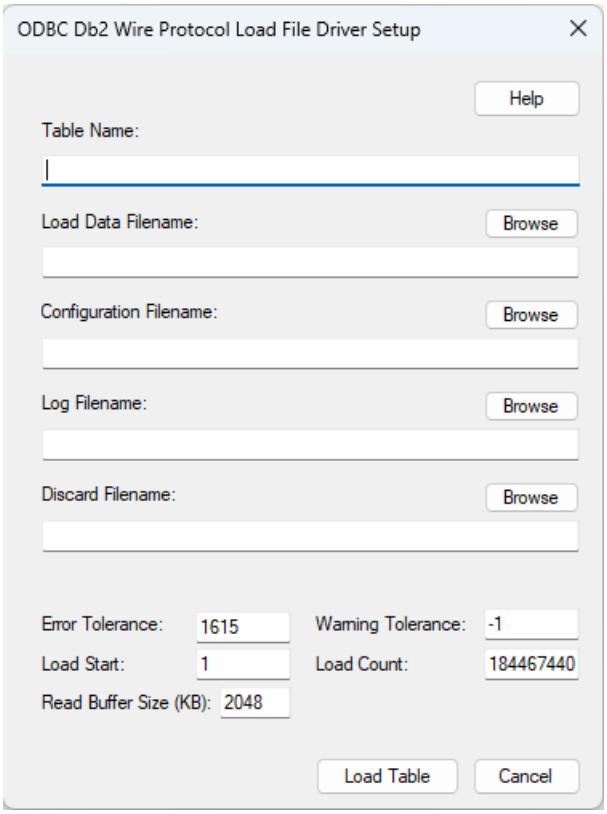
  

- **Table Name**: A string that specifies the name of the target database table into which the data is to be loaded.
- **Configuration Filename**: A string that specifies the path (relative or absolute) and file name of the bulk configuration file. The file name must be the fully qualified path to the configuration file.

- Click **Verify** to verify table structure or click **Cancel**.

2. To bulk load data from the bulk data file to a database table, click **Load Table**from the Bulk tab. The Load File dialog box appears.The load operation can create a log file and can also create a discard file that contains rows rejected during the load. The discard file is in the same format as the bulk load data file. After fixing reported issues in the discard file, the bulk load can be reissued using the discard file as the bulk load data file.

Figure 10: Load File dialog box



The export operation can be configured such that if any errors or warnings occur:

- The operation always completes.
- The operation always terminates.
- The operation terminates after a certain threshold of warnings or errors is exceeded.

If a load fails, the Load Start and Load Count options can be used to control which rows are loaded when a load is restarted after a failure.

The following fields are available on the Load File dialog box.

- **Table Name**: A string that specifies the name of the target database table into which the data is loaded.
- **Load Data Filename**: A string that specifies the path (relative or absolute) and file name of the bulk data file from which the data is loaded. The file name must be the fully qualified path to the bulk data file.
- **Configuration Filename**: A string that specifies the path (relative or absolute) and file name of the bulk configuration file. The file name must be the fully qualified path to the configuration file.
- If you do not specify a value for Configuration Filename, the bulk configuration file name is assumed to be `bulk_data_file_name`.xml.

- **Log Filename**: A string that specifies the path (relative or absolute) and file name of the bulk log file. The file name must be the fully qualified path to the log file. Specifying a value for Log Filename creates the file if it does not already exist. Events logged to this file are:
  - Total number of rows read
  - Message for each row that failed to load
  - Total number of rows that failed to load
  - Total number of rows successfully loaded

  Information about the load is written to this file, preceded by a header. Information about the next load is appended to the end of the file.

  If you do not specify a value for Log Filename, no log file is created.
- **Discard Filename**: A string that specifies the path (relative or absolute) and file name of the bulk discard file. The file name must be the fully qualified path to the discard file. Any row that cannot be inserted into database as result of bulk load is added to this file, with the last row rejected added to the end of the file.

  Information about the load is written to this file, preceded by a header. Information about the next load is appended to the end of the file.

  If you do not specify a value for Discard Filename, a discard file is not created.
- **Error Tolerance**: A value that specifies the number of errors to tolerate before an operation terminates. A value of `0` indicates that no errors are tolerated; the operation fails when the first error is encountered.

  The default of `-1` means that an infinite number of errors is tolerated.
- **Load Start**: A value that specifies the first row to be loaded from the data file. Rows are numbered starting with 1. For example, when Load Start is 10, the first 9 rows of the file are skipped and the first row loaded is row 10. This option can be used to restart a load after a failure.

  The default value is `1`.
- **Read Buffer Size (KB)**: A value that specifies the size, in KB, of the buffer that is used to read the bulk data file for a bulk load operation.

  The default value is `2048`.
- **Warning Tolerance**: A value that specifies the number of warnings to tolerate before an operation terminates. A value of `0` indicates that no warnings are tolerated; the operation fails when the first warning is encountered.

  The default of `-1` means that an infinite number of warnings is tolerated.
- **Load Count**: A value that specifies the number of rows to be loaded from the data file. The bulk load operation loads rows up to the value of Load Count from the file to the database. It is valid for Load Count to specify more rows than exist in the data file. The bulk load operation completes successfully when either the number of rows specified by the Load Count value has been loaded or the end of the data file is reached. This option can be used in conjunction with Load Start to restart a load after a failure.

  The default value is the maximum value for SQLULEN. If set to `0`, no rows are loaded.
- Click **Load Table** to connect to the database and load the table or click **Cancel**.

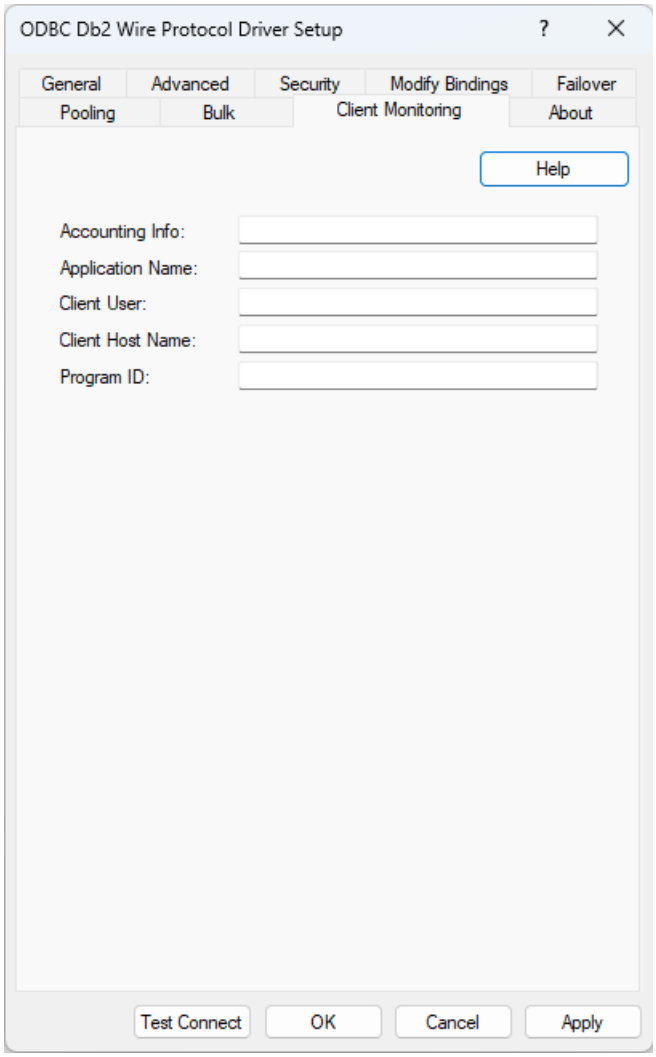
See also

Configuring the driver using the GUI on page 43

Client Monitoring tab

The Client Monitoring 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 11: Client Monitoring tab



Connection Options: Client Monitoring	Default
Accounting Info on page 117	No default value
Application Name on page 119	No default value
Client User on page 126	No default value

Connection Options: Client Monitoring	Default
Client Host Name on page 125	No default value
Program ID on page 159	No default value

See also

Configuring the driver using the GUI on page 43

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=Db2ACCOUNT;DB=Db2DATA;UID=JOHN;PWD=XYZZY
```

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

```
FILEDSN=Db2.dsn;DB=Db2DATA;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=123.456.78.90;
PORT=5179;DB=Db2DATA;UID=JOHN;PWD=XYZZY
```

See also

Connection option descriptions on page 109

Connection string examples on page 18

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:

```
Db2=DataDirect 8.0 Db2 Wire Protocol
```

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

The `[ODBC Data Sources]` section also includes data source definitions. The default `odbc.ini` contains a data source definition for the driver. Each data source definition begins with a data source name in square brackets, for example, `[Db2 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 109
Sample default odbc.ini file on page 62
File data sources on page 66
IANAAppCodePage on page 147
DSN-less connections on page 64

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]
Db2=DataDirect 8.0 Db2 Wire Protocol

[Db2]
Driver=ODBCHOME/lib/ivdb228.so
Description=DataDirect 8.0 Db2 Wire Protocol Driver
AccountingInfo=
AddStringToCreateTable=
AlternateID=
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=0
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
```

```
BulkLoadFieldDelimiter=
BulkLoadRecordDelimiter=
CatalogSchema=
CharsetFor65535=0
ClientHostName=
ClientUser=
Collection=
ConcurrentAccessResolution=0
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CryptoLibName=
CryptoProtocolVersion=TLSv1.3,TLSv1.2
CurrentFunctionPath=
#Database applies to Db2 UDB only
Database=<database_name>
DefaultIsolationLevel=1
DynamicSections=1000
EnableFIPS=1
EnableBulkLoad=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
GrantAuthid=PUBLIC
GrantExecute=1
GSSClient=native
HostNameInCertificate=
IpAddress=<Db2_server_host>
KeepAlive=0
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
Location=<location_name>
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinLongVarcharSize=
MinPoolSize=0
PackageCollection=NULLID
PackageNamePrefix=DD
PackageOwner=
Password=
Pooling=0
ProgramID=
QueryTimeout=0
ReportCodePageConversionErrors=0
SSLLibName=
TcpPort=50000
TrustStore=
TrustStorePassword=
UseCurrentSchema=1
ValidateServerCertificate=1
VarcharThreshold=
WithHold=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, `[My Datasource]`.
  d) Modify the attributes in the new definition as necessary based on your system specifics.

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

  e) In the `[ODBC]` section at the beginning of the file, add a new *data_source_name=installed-driver* pair containing the new data source name and the appropriate installed driver name.
  f) After making all modifications, save the `odbc.ini` file and close the text editor.

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

See also

Connection option descriptions on page 109

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 Db2 Wire Protocol=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 41
Configuration through the system information (odbc.ini) file on page 61

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 Db2 Wire Protocol=Installed

[DataDirect 8.0 Db2 Wire Protocol]
Driver=ODBCHOME/lib/ivdb228.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, either Windows UNIX, or Linux. See "Configuring and connecting to data sources" for a general description of ODBC data sources on both Windows, UNIX, and Linux.

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

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

```
[ODBC]
Driver=DataDirect 8.0 Db2 Wire Protocol
...
IPAddress=123.456.78.90
...
TCPPort=50000
...
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\Db2.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/Db2.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/Db2.dsn;DB=Db2DATA;UID=JOHN;PWD=XYZZY
```

See also

Configuring and connecting to data sources on page 39

DSN-less connections on page 64
Additional configuration methods for UNIX and Linux on page 61

Password Encryption Tool (UNIX/Linux only)

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

- KeyPassword
- KeyStorePassword
- TrustStorePassword
- Password

To use the Password Encryption Tool:

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

   `ddencpwd` *`password`*

   where:

   `password`

   is the password you want to encrypt.

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

   `KeyPassword=`*`returned_value`*

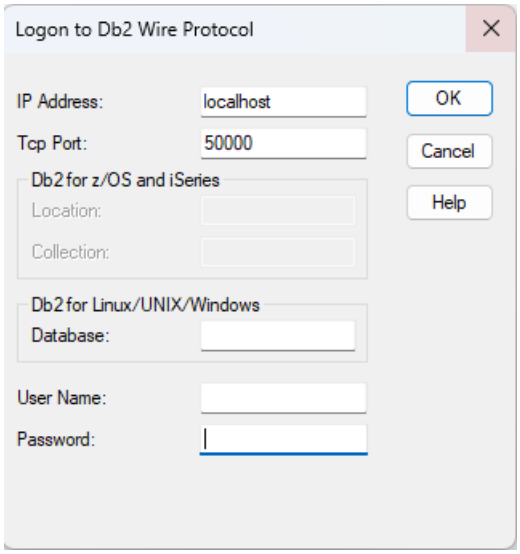
4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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

Using a logon dialog box

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

Figure 12: Logon to Db2 dialog box



In this dialog box, provide the following information:

1. In the IP Address field, type either the host name or the IP address of the machine where catalog tables are stored.
2. In the TCP Port field, type the port number that is assigned to the Db2 DRDA listener process on the server host machine. The default is `50000`.
3. Provide information in one of the following sets of fields based on the server to which you want to connect:
   - **For Db2 for z/OS and iSeries**:
     - In the Location field, type the name of the Db2 location that you want to access.
     - In the Collection field, type the name of the current collection or library.
   - **For Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud**:
     - In the Database field, type the name of the database to which you want to connect.
4. In the User Name field, type your Db2 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 authentication* authenticates using the specified user IDs and passwords.
- *Client authentication* uses the user ID and password of the user logged onto the system on which the driver is running to authenticate the user to the database. The database server relies on the client to authenticate the user and does not provide additional authentication.
- *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.

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

See also

Authentication Method on page 121

User ID/password authentication

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

Important: The procedure given in this section uses the user ID/password authentication without encryption. To use it with encryption, set the Authentication Method option to one of the following values: `1` (Encrypt Password), `2` (Encrypt UID and Password), `7` (Encrypted Password AES), and `8` (Encrypted UID and Password AES). See "Authentication Method" for details.

To configure the driver to use user ID/password (no encryption) authentication:

- Set the IP Address (IPAddress) option to specify the host name or the IP address of the machine where catalog tables are stored.
- Set the TCP Port (TCPPort) option to specify the port number that is assigned to the Db2 DRDA listener process on the server host machine.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.
- Set the Location Name (Location) option to specify the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.
- Set the Collection option to specify the current collection or library. Valid only on Db2 for z/OS and Db2 for i.
- 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 connect to Db2 for Linux, UNIX, and Windows using user ID/password (no encryption) authentication.

Connection string

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=localhost;TCPPort=50000;
Database=db2data;LogonID=jsmith;Password=secret;
```

odbc.ini

```
[Db2]
Driver=ODBCHOME/lib/xxdb228.yy
...
IPAddress=localhost
...
TCPPort=50000
...
Database=db2data
...
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.

See also

Connection option descriptions on page 109

Kerberos authentication

If you are using Kerberos, verify that your environment meets the requirements listed in the following table before you configure the driver for Kerberos authentication.

Table 4: Kerberos Authentication Requirements for the Db2 Wire Protocol Driver

Component	Requirements
Database server	The database server must be running one of the following database versions: • Db2 V8.1 or higher for Linux/UNIX/Windows • Db2 V8.x or higher for z/OS
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by one of the following methods: • Windows Active Directory • MIT Kerberos 1.4.2 or higher

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.

UNIX® 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 IP Address (IPAddress) option to specify the host name or the IP address of the machine where catalog tables are stored.
- Set the TCP Port (TCPPort) option to specify the port number that is assigned to the Db2 DRDA listener process on the server host machine.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows.
- Set the name of the Db2 location that you want to access. Valid only on Db2 for z/OS.
- Set the current collection or library. Valid only on Db2 for z/OS.
- 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).

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

Connection string

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;AuthenticationMethod=4;
IPAddress=123.456.78.90;TCPPort=50000;Database=Payroll;GSSClient=gss123;
```

odbc.ini

```
[Db2]
Driver=ODBCHOME/lib/xxdb228.yy
...
AuthenticationMethod=4
...
IPAddress=123.456.78.90
...
TCPPort=50000
...
Database=Payroll
...
```

```
GSSClient=gss123
...
```

See also

Connection option descriptions on page 109

TLS/SSL encryption

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

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

The handshake involves the following types of authentication:

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

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

Certificates

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

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

TLS/SSL server authentication

When the client makes a connection request, the server presents its certificate for the client to accept or deny. The client checks the issuer of the certificate against a list of trusted Certificate Authorities (CAs) whose root certificates reside in one or both of the following stores on the client:

- On Windows operating systems: A permanent storage known as *Windows certificate store*. To learn how to import the required root certificates into the Windows certificate store, see "Importing root certificates into the Windows certificate store."
- On both Windows and non-Windows operating systems: An encrypted file known as *truststore file*. Most truststore files are password-protected. The driver must be able to locate the truststore file and unlock it with the appropriate password. Two connection options are available to the driver to provide this information: Trust Store (Truststore) and Trust Store Password (TruststorePassword).

If the server certificate matches a root certificate in either of the stores, an encrypted connection is established between the client and the server. If the certificate does not match, the connection fails and the client generates an error.

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

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

- Set the IP Address (IPAddress) option to specify the host name or the IP address of the machine where catalog tables are stored.
- Set the TCP Port (TCPPort) option to specify the port number that is assigned to the Db2 DRDA listener process on the server host machine.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.
- Set the Location Name (Location) option to specify the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.
- Set the Collection option to specify the current collection or library. Valid only on Db2 for z/OS and Db2 for i.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Trust Store (Truststore) option to specify either the full path of the truststore file or the contents of the TLS/SSL certificates.

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 root 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".

- Set the Truststore Password (TruststorePassword) option to specify the password that is used to access the truststore file.
- Optionally, set the Enable FIPS (EnableFIPS) connection option to `1` to allow the driver to load the FIPS provider. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2. If you do not specify a value for Enable FIPS, the driver uses its default value (`0`) and loads the default provider.

Note:

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
- Do not set the Truststore Password connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms" for more information.

The following examples configure the driver to use data encryption via the TLS/SSL server authentication. In these configurations, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate, and loads the FIPS provider for data encryption. In addition, the driver uses user ID/password authentication to connect to Db2 for Linux, UNIX, and Windows.

Connection string

Truststore:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EnableFIPS=1;
EncryptionMethod=1;IPAddress=localhost;TCPPort=50000;Database=db2data;
HostNameInCertificate=MySubjectAltName;Truststore=TrustStoreName;
ValidateServerCertificate=1;
```

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

Windows certificate store:

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EnableFIPS=1;
EncryptionMethod=1;IPAddress=localhost;TCPPort=50000;Database=db2data;
HostNameInCertificate=MySubjectAltName;ValidateServerCertificate=1;
```

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

odbc.ini

Truststore:

```
Driver=ODBCHOME/lib/ivdb2XX.so
Description=DataDirect Db2 Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
IPAddress=localhost;
...
HostNameInCertificate=MySubjectAltName
...
TCPPort=50000;
...
Database=db2data;
...
Truststore=TrustStoreName
...
ValidateServerCertificate=1
...
```

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

Windows certificate store:

```
Driver=ODBCHOME/lib/ivdb2xx.so
Description=DataDirect Db2 Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
IPAddress=localhost;
...
HostNameInCertificate=MySubjectAltName
...
TCPPort=50000;
...
Database=db2data;
...
ValidateServerCertificate=1
...
```

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

See also

Importing root certificates into the Windows certificate store on page 77
Trust Store on page 165
Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 76
Connection option descriptions on page 109
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 80

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 (Truststore) connection option as well. However, if it is specified using both Trust Store 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=Db2WP_SSL;UID=jsmith;PWD=secret", SQL_NTS,
NULL, 0, NULL, SQL_DRIVER_NOPROMPT);
```

See also

Trust Store on page 165

Importing root certificates into the Windows certificate store

This section provides you with an overview of the steps required to import the required root certificates from a truststore file to the Windows certificate store.

You can import root certificates using either the Certificate Import Wizard or a PowerShell script.

Importing root certificates using Certificate Import Wizard

To import root certificates using Certificate Import Wizard:

1. Double-click the trustore file. The **Certificate Import Wizard** window appears.
2. Select the **Current User** radio button; then, click **Next**.
3. Verify the file path and name available in the **File name** field; then, click **Next**.
4. Enter the password to unlock the truststore file; then, click **Next**.
5. Select the **Automatically select the certificate store based on the type of certificate** radio button; then, click **Next**.
6. Click **Finish**.

The root certificates are imported into the following location in the Windows certificate store: **Certificates** > **Trusted Root Certification Authorities** > **Certificates**.

Note: At times, Windows doesn't trust the imported certificates and imports them into **Certificates** > **Intermediate Certificate Authorities** > **Certificates**. In such cases, manually copy the imported certificates from **Intermediate Certificate Authorities** to **Trusted Root Certification Authorities**.

Importing root certificates using a PowerShell script

To import root certificates using a PowerShell script:

1. Open PowerShell in Administrator mode.
2. Type the following command; then, press **ENTER**.

```
Import-PfxCertificate -Password (ConvertTo-SecureString -String "truststore_password"
 -AsPlainText -Force) -CertStoreLocation Cert:\LocalMachine\Root -FilePath
truststore_filepath
```

where:

`truststore_password`

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

`truststore_filepath`

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

The root certificates are imported into the following location in the Windows certificate store: **Certificates** > **Trusted Root Certification Authorities** > **Certificates**.

Note: At times, Windows doesn't trust the imported certificates and imports them into **Certificates** > **Intermediate Certificate Authorities** > **Certificates**. In such cases, manually copy the imported certificates from **Intermediate Certificate Authorities** to **Trusted Root Certification Authorities**.

TLS/SSL client authentication

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

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

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

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

- Set the IP Address (IPAddress) option to specify the host name or the IP address of the machine where catalog tables are stored.
- Set the TCP Port (TCPPort) option to specify the port number that is assigned to the Db2 DRDA listener process on the server host machine.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect. Valid only on Db2 for Linux, UNIX, and Windows; Db2 Hosted; and Db2 Warehouse on Cloud.
- Set the Location Name (Location) option to specify the name of the Db2 location that you want to access. Valid only on Db2 for z/OS and Db2 for i.
- Set the Collection option to specify the current collection or library. Valid only on Db2 for z/OS and Db2 for i.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Key Store (Keystore) option to specify the location of the keystore file.
- Set the Keystore Password (KeystorePassword) option to specify the password that is used to access the keystore file.
- Optionally, set the Enable FIPS (EnableFIPS) connection option to `1` to allow the driver to load the FIPS provider. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2. If you do not specify a value for Enable FIPS, the driver uses its default value (`0`) and loads the default provider.

Note:

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit.
- For using the FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms" for more information.
- Do not set the Keystore Password connection option when using the FIPS provider. The keystore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.

The following examples configure the driver to use data encryption via the SSL client authentication. In these configurations, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate, and loads the FIPS provider for data encryption. In addition, the driver uses user ID/password authentication to connect to Db2 for Linux, UNIX, and Windows.

Connection string

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;EnableFIPS=1;
EncryptionMethod=1;IPAddress=localhost;HostNameInCertificate=MySubjectAltName;
TCPPort=50000;Database=db2data;Keystore=KeyStoreName;ValidateServerCertificate=1;
```

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

odbc.ini

```
Driver=ODBCHOME/lib/ivdb2xx.so
Description=DataDirect Db2 Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
IPAddress=localhost
...
TCPPort=50000
...
HostNameInCertificate=MySubjectAltName
...
Database=db2data;
...
KeyStore=KeyStoreName
...
ValidateServerCertificate=1
...
```

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

See also

Connection option descriptions on page 109
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 80

Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms

For using the OpenSSL 3.5 providers (FIPS and default), the certificates for TLS/SSL encryption must be generated using the OpenSSL 3.5-compliant cryptographic algorithms.

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

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

Note: OpenSSL 3.5.x enforces Security Level 2, which requires all RSA/DSA keys to be at least 2048 bits. To meet these security requirements, certificates must be updated to use RSA keys of 2048 bits or higher. Any certificates that still use 1024-bit keys will be rejected during the SSL/TLS handshake.

For truststore.pfx, every CA certificate must use a 2048-bit or larger public key.

For keystore.pfx, both the private key and the corresponding certificate must be 2048 bits or greater to comply with OpenSSL Security Level 2.

Truststore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`truststore_filename`

is the name of the truststore file.

`cryptographic_algorithm`

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

`truststore_password`

is the password required for accessing the truststore file.

Example:

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

Keystore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`privatekey_file`

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

`truststore_filename`

is the name of the keystore file.

`cryptographic_algorithm`

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

Example:

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

Note: If you are using the Windows certificate store for TLS/SSL encryption, import the certificates generated with the OpenSSL 3.5-compliant algorithms into the store.

Designating an OpenSSL library

Important: Currently, the driver supports version 3.5.6 of the OpenSSL library by default.

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

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

File replacement

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

Windows:

- Version 3.5: `ivopenssl.dll` and `ddopenssl.dll`

UNIX/Linux:

- Version 3.5: `ivopenssl.so` and `ddopenssl.so`

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.

Note: The CryptoLibName and SSLLibName options must be configured if you are using OpenSSL version 3.0.

See also

CryptoLibName on page 132
SSLLibName on page 163

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 83
Guidelines for primary and alternate servers on page 85

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.

The following examples show how to configure the driver to use connection failover in conjunction with some of its optional features when connecting to the Db2 for z/OS and iSeries servers. These examples use the user ID/password (No Encryption) authentication method for authentication.

Connection string

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;
AlternateServers=(IPAddress=localhost:TCPPort=50000:Location=NCASV52:Collection=payroll,
IPAddress=123.456.78.90:TCPPort=50001:Location=NCASV52:Collection=accounting);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0;
LogonID=jsmith;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.

odbc.ini file

```
Driver=ODBCHOME/lib/ivdb2xx.so
Description=DataDirect Db2 Wire Protocol
...
AlternateServers=(IPAddress=localhost:TCPPort=50000:Location=NCASV52:Collection=payroll,
IPAddress=123.456.78.90:TCPPort=50001:Location=NCASV52:Collection=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 109

Guidelines for primary and alternate servers

To ensure that failover works correctly, use the following guidelines for primary and alternate servers:

- Alternate servers should mirror data on the primary server or be part of a configuration where multiple database nodes share the same physical data.
- If using failover with Db2 HADR, the primary server must be the primary server configured in your HADR system and any alternate server must be a standby server configured in your HADR system.

DataDirect Bulk Load

UNIX®

Supported on Windows, UNIX, and Linux only.

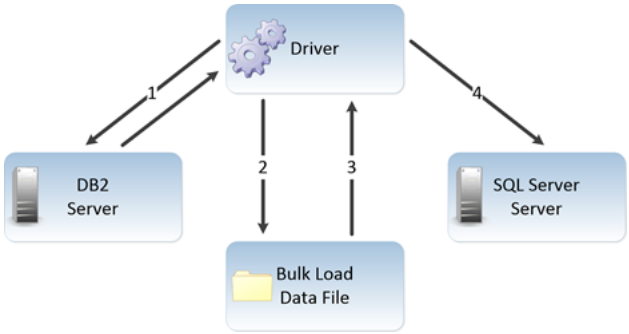
On Windows, UNIX, and Linux, the driver supports DataDirect Bulk Load, a feature that allows your application to send large numbers of rows of data to a database. Since Db2 does not have native bulk load support, the driver supports bulk through the native parameter array mechanism.

DataDirect Bulk Load requires a licensed installation of the driver. If the driver is installed with an evaluation license, the bulk load feature is available for prototyping with your applications, but with limited scope. Contact your sales representative or Progress DataDirect SupportLink for further information.

Because a bulk load operation may bypass data integrity checks, your application must ensure that the data it is transferring does not violate integrity constraints in the database. For example, suppose you are bulk loading data into a database table and some of that data duplicates data stored as a primary key, which must be unique. The driver will not throw an exception to alert you to the error; your application must provide its own data integrity checks.

Bulk load operations are accomplished by exporting the results of a query from a database into a comma-separated value (CSV) file, a bulk load data file. The driver then loads the data from bulk load data file into a different database. The file can be used by any DataDirect *for* ODBC driver. In addition, the bulk load data file is supported by other DataDirect product lines that feature bulk loading, for example, a DataDirect Connect for ADO.NET data provider that supports bulk load.

Suppose that you had customer data on a Db2 server and need to export it to a SQL Server server. The driver would perform the following steps:



1. Application using Db2 Wire Protocol driver sends query to and receives results from Db2 server.
2. Driver exports results to bulk load data file.
3. Driver retrieves results from bulk load data file.
4. Driver bulk loads results on SQL Server server.

Bulk Export and Load Methods

You can take advantage of DataDirect Bulk Load either through the Driver setup dialog or programmatically.

Applications that are already coded to use parameter array batch functionality can leverage DataDirect Bulk Load features through the Enable Bulk Load connection option on the Bulk tab of the Driver setup dialog. Enabling this option automatically converts the parameter array batch operation to use the database bulk load protocol without any code changes to your application.

If you are not using parameter array batch functionality, the bulk operation buttons **Export Table** and **Load Table** on the Bulk tab of the driver Setup dialog also allow you to use bulk load functionality without any code changes. See "Bulk tab" for a description of the Bulk tab.

If you want to integrate bulk load functionality seamlessly into your application, you can include code to use the bulk load functions exposed by the driver.

For your applications to use DataDirect Bulk Load functionality, they must obtain driver connection handles and function pointers, as follows:

1. Use SQLGetInfo with the parameter SQL_DRIVER_HDBC to obtain the driver's connection handle from the Driver Manager.
2. Use SQLGetInfo with the parameter SQL_DRIVER_HLIB to obtain the driver's shared library or DLL handle from the Driver Manager.
3. Obtain function pointers to the bulk load functions using the function name resolution method specific to your operating system. The `bulk.c` example program shipped with the drivers contains the function resolveName that illustrates how to obtain function pointers to the bulk load functions.

This is detailed in the code samples that follow.

See also

Bulk tab on page 54

Exporting data from a database

You can export data from a database in one of three ways:

- From a table by using the driver Setup dialog
- From a table by using DataDirect functions
- From a result set by using DataDirect statement attributes

From the DataDirect driver Setup dialog, select the **Bulk** tab and click **Export Table**. See the driver configuration chapter for a description of this procedure.

Your application can export a table using the DataDirect functions ExportTableToFile (ANSI application) or ExportTableToFileW (Unicode application). The application must first obtain driver connection handles and function pointers, as shown in the following example:

```
HDBC      hdbc;
HENV      henv;
void      *driverHandle;
HMODULE      hmod;
PExportTableToFile exportTableToFile;

char      tableName[128];
char      fileName[512];
char      logFile[512];
int       errorTolerance;
int       warningTolerance;
int       codePage;

/* Get the driver's connection handle from the DM.
   This handle must be used when calling directly into the driver. */

rc = SQLGetInfo (hdbc, SQL_DRIVER_HDBC, &driverHandle, 0, NULL);
if (rc != SQL_SUCCESS) {
    ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
    EnvClose (henv, hdbc);
    exit (255);
}

/* Get the DM's shared library or DLL handle to the driver. */

rc = SQLGetInfo (hdbc, SQL_DRIVER_HLIB, &hmod, 0, NULL);
if (rc != SQL_SUCCESS) {
    ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
    EnvClose (henv, hdbc);
    exit (255);
}
```

```
    exportTableToFile = (PExportTableToFile)
    resolveName (hmod, "ExportTableToFile");
if (! exportTableToFile) {
    printf ("Cannot find ExportTableToFile!\n");
    exit (255);
}

rc = (*exportTableToFile) (
      driverHandle,
      (const SQLCHAR *) tableName,
      (const SQLCHAR *) fileName,
      codePage,
      errorTolerance, warningTolerance,
      (const SQLCHAR *) logFile);
if (rc == SQL_SUCCESS) {
      printf ("Export succeeded.\n");
}
else {
      driverError (driverHandle, hmod);
}
```

Your application can export a result set using the DataDirect statement attributes SQL_BULK_EXPORT and SQL_BULK_EXPORT_PARAMS.

The export operation creates a bulk load data file with a .csv extension in which the exported data is stored. For example, assume that a source table named GBMAXTABLE contains four columns. The resulting bulk load data file GBMAXTABLE.csv containing the results of a query would be similar to the following:

```
1,0x6263,"bc","bc"
2,0x636465,"cde","cde"
3,0x64656667,"defg","defg"
4,0x6566676869,"efghi","efghi"
5,0x666768696a6b,"fghijk","fghijk"
6,0x6768696a6b6c6d,"ghijklm","ghijklm"
7,0x68696a6b6c6d6e6f,"hijklmno","hijklmno"
8,0x696a6b6c6d6e6f7071,"ijklmnopq","ijklmnopq"
9,0x6a6b6c6d6e6f70717273,"jklmnopqrs","jklmnopqrs"
10,0x6b,"k","k"
```

A bulk load configuration file with and .xml extension is also created when either a table or a result set is exported to a bulk load data file. See "The bulk load configuration file" for an example of a bulk load configuration file.

In addition, a log file of events as well as external overflow files can be created during a bulk export operation. The log file is configured through either the driver Setup dialog Bulk tab, the ExportTableToFile function, or the SQL_BULK_EXPORT statement attribute. The external overflow files are configured through connection options; see "External overflow files" for details.

See also

The bulk load configuration file on page 90
External overflow files on page 93

Bulk loading to a database

You can load data from the bulk load data file into the target database through the DataDirect driver Setup dialog by selecting the Bulk tab and clicking **Load Table**. See "Bulk tab" for a description of this procedure.

Your application can also load data from the bulk load data file into the target database using the using the DataDirect functions LoadTableFromFile (ANSI application) or LoadTableFromFileW (Unicode application). The application must first obtain driver connection handles and function pointers, as shown in the following example:

```
HDBC      hdbc;
HENV      henv;
void      *driverHandle;
HMODULE   hmod;
PLoadTableFromFile loadTableFromFile;
char      tableName[128];
char      fileName[512];
char      configFile[512];
char      logFile[512];
char      discardFile[512];
int       errorTolerance;
int       warningTolerance;
int       loadStart;
int       loadCount;
int       readBufferSize;

/* Get the driver's connection handle from the DM.
   This handle must be used when calling directly into the driver.*/

rc = SQLGetInfo (hdbc, SQL_DRIVER_HDBC, &driverHandle, 0, NULL);
if (rc != SQL_SUCCESS) {
  ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
  EnvClose (henv, hdbc);
  exit (255);
}
/* Get the DM's shared library or DLL handle to the driver. */

rc = SQLGetInfo (hdbc, SQL_DRIVER_HLIB, &hmod, 0, NULL);
if (rc != SQL_SUCCESS) {
  ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
  EnvClose (henv, hdbc);
  exit (255);
}

loadTableFromFile = (PLoadTableFromFile)
  resolveName (hmod, "LoadTableFromFile");
if (! loadTableFromFile) {
  printf ("Cannot find LoadTableFromFile!\n");
  exit (255);
}

rc = (*loadTableFromFile) (
     driverHandle,
     (const SQLCHAR *) tableName,
     (const SQLCHAR *) fileName,
     errorTolerance, warningTolerance,
     (const SQLCHAR *) configFile,
     (const SQLCHAR *) logFile,
     (const SQLCHAR *) discardFile,
     loadStart, loadCount,
     readBufferSize);
if (rc == SQL_SUCCESS) {
     printf ("Load succeeded.\n");
}
else {
     driverError (driverHandle, hmod);
}
```

Refer to "DataDirect Bulk Load functions" in the *Progress DataDirect for ODBC Drivers Reference* for more information on supported functions.

Use the BulkLoadBatchSize connection attribute to specify the number of rows the driver loads to the data source at a time when bulk loading data. Performance can be improved by increasing the number of rows the driver loads at a time because fewer network round trips are required. Be aware that increasing the number of rows that are loaded also causes the driver to consume more memory on the client.

A log file of events as well as a discard file that contains rows rejected during the load can be created during a bulk load operation. These files are configured through either the driver Setup dialog Bulk tab or the LoadTableFromFile function.

The discard file is in the same format as the bulk load data file. After fixing reported issues in the discard file, the bulk load can be reissued using the discard file as the bulk load data file.

See also

Bulk tab on page 54

The bulk load configuration file

A bulk load configuration file is created when either a table or a result set is exported to a bulk load data file. This file has the same name as the bulk load data file, but with an `.xml` extension.

The bulk load configuration file defines in its metadata the names and data types of the columns in the bulk load data file. The file defines these names and data types based on the table or result set created by the query that exported the data.

It also defines other data properties, such as length for character and binary data types, the character encoding code page for character types, precision and scale for numeric types, and nullability for all types.

When a bulk load data file cannot read its configuration file, the following defaults are assumed:

- All data is read in as character data. Each value between commas is read as character data.
- The default character set is defined, on Windows, by the current Windows code page. On UNIX/Linux, it is the IANAAppCodePage value, which defaults to `4`.

For example, the format of the bulk load data file `GBMAXTABLE.csv` (discussed in "Exporting data from a database") is defined by the bulk load configuration file, `GBMAXTABLE.xml`, as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<table codepage="UTF-16LE" xsi:noNamespaceSchemaLocation=
"http://media.datadirect.com/download/docs/ns/bulk/BulkData.xsd" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <column datatype="DECIMAL" precision="38" scale="0" nullable=
      "false">INTEGERCOL</column>
    <column datatype="VARBINARY" length="10" nullable=
      "true">VARBINCOL</column>
    <column datatype="VARCHAR" length="10" sourcecodepage="Windows-1252"
      externalfilecodepage="Windows-1252" nullable="true">VCHARCOL</column>
    <column datatype="VARCHAR" length="10" sourcecodepage="Windows-1252"
      externalfilecodepage="Windows-1252" nullable="true">UNIVCHARCOL</column>
  </row>
</table>
```

See also

Exporting data from a database on page 87

Bulk load configuration file schema

The bulk load configuration file is supported by an underlying XML Schema defined at:

http://media.datadirect.com/download/docs/ns/bulk/BulkData.xsd

The bulk load configuration file must conform to the bulk load configuration XML schema. Each bulk export operation generates a bulk load configuration file in UTF-8 format. If the bulk load data file cannot be created or does not comply with the XML Schema described in the bulk load configuration file, an error is generated.

Verification of the bulk load configuration file

You can verify the metadata in the configuration file against the data structure of the target database table. This insures that the data in the bulk load data file is compatible with the target database table structure.

The verification does not check the actual data in the bulk load data file, so it is possible that the load can fail even though the verification succeeds. For example, if you were to update the bulk load data file manually such that it has values that exceed the maximum column length of a character column in the target table, the load would fail.

Not all of the error messages or warnings that are generated by verification necessarily mean that the load will fail. Many of the messages simply notify you about possible incompatibilities between the source and target tables. For example, if the bulk load data file has a column that is defined as an integer and the column in the target table is defined as smallint, the load may still succeed if the values in the source column are small enough that they fit in a smallint column.

To verify the metadata in the bulk load configuration file through the DataDirect driver Setup dialog, select the Bulk tab and click **Verify**. See "Bulk tab" for a description of this procedure.

Your application can also verify the metadata of the bulk load configuration file using the DataDirect functions ValidateTableFromFile (ANSI application) or ValidateTableFromFileW (Unicode application). The application must first obtain driver connection handles and function pointers, as shown in the following example:

```
HDBC      hdbc;
HENV      henv;
void      *driverHandle;
HMODULE   hmod;
PValidateTableFromFile validateTableFromFile;
char      tableName[128];
char      configFile[512];
char      messageList[10240];
SQLLEN    numMessages;
/* Get the driver's connection handle from the DM.
   This handle must be used when calling directly into the driver. */
rc = SQLGetInfo (hdbc, SQL_DRIVER_HDBC, &driverHandle, 0, NULL);
if (rc != SQL_SUCCESS) {
    ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
    EnvClose (henv, hdbc);
    exit (255);
}/* Get the DM's shared library or DLL handle to the driver. */
rc = SQLGetInfo (hdbc, SQL_DRIVER_HLIB, &hmod, 0, NULL);
if (rc != SQL_SUCCESS) {
    ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
    EnvClose (henv, hdbc);
    exit (255);
}
validateTableFromFile = (PValidateTableFromFile)
  resolveName (hmod, "ValidateTableFromFile");
if (!validateTableFromFile) {
  printf ("Cannot find ValidateTableFromFile!\n");
  exit (255);
}
messageList[0] = 0;
numMessages = 0;
rc = (*validateTableFromFile) (
      driverHandle,
      (const SQLCHAR *) tableName,
      (const SQLCHAR *) configFile,
      (SQLCHAR *) messageList,
      sizeof (messageList),
      &numMessages);
printf ("%d message%s%s\n", numMessages,
       (numMessages == 0) ? "s" :
       ((numMessages == 1) ? " : " : "s : "),
       (numMessages > 0) ? messageList : "");
if (rc == SQL_SUCCESS) {
       printf ("Validate succeeded.\n");
}else {
    driverError (driverHandle, hmod);
}
```

See also

Bulk tab on page 54

Sample applications

Progress DataDirect provides a sample application that demonstrates the bulk export, verification, and bulk load operations. This application is located in the `\samples\bulk` subdirectory of the product installation directory along with a text file named `bulk.txt`. Please consult `bulk.txt` for instructions on using the sample bulk load application.

A bulk streaming application is also provided in the `\samples\bulkstrm` subdirectory along with a text file named `bulkstrm.txt`. Please consult `bulkstrm.txt` for instructions on using the bulk streaming application.

Character set conversions

It is most performance-efficient to transfer data between databases that use the same character sets. At times, however, you might need to bulk load data between databases that use different character sets. You can do this by choosing a character set for the bulk load data file that will accommodate all data. If the source table contains character data that uses different character sets, then one of the Unicode character sets, UTF-8, UTF-16BE, or UTF-16LE must be specified for the bulk load data file. A Unicode character set should also be specified in the case of a target table uses a different character set than the source table to minimize conversion errors. If the source and target tables use the same character set, that set should be specified for the bulk load data file.

A character set is defined by a code page. The code page for the bulk load data file is defined in the configuration file and is specified through either the Code Page option of the Export Table driver Setup dialog or through the IANAAppCodePage parameter of the ExportTableToFile function.

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

Any character conversion errors are handled based on the value of the Report Codepage Conversion Errors connection option. See the individual driver chapters for a description of this option.

The configuration file may optionally define a second code page value for each character column (`externalfilecodepage`). If character data is stored in an external overflow file (see "External overflow files"), this second code page value is used for the external file.

See also

External overflow files on page 93

External overflow files

In addition to the bulk load data file, DataDirect Bulk Load can store bulk data in external overflow files. These overflow files are located in the same directory as the bulk load data file. Different files are used for binary data and character data. Whether or not to use external overflow files is a performance consideration. For example, binary data is stored as hexadecimal-encoded character strings in the main bulk load data file, which increases the size of the file per unit of data stored. External files do not store binary data as hex character strings, and, therefore, require less space. On the other hand, more overhead is required to access external files than to access a single bulk load data file, so each bulk load situation must be considered individually.

The value of the Bulk Binary Threshold connection option determines the threshold, in KB, over which binary data is stored in external files instead of in the bulk load data file. Likewise, the Bulk Character Threshold connection option determines the threshold for character data.

In the case of an external character data file, the character set of the file is governed by the bulk load configuration file. If the bulk load data file is Unicode and the maximum character size of the source data is 1, then the data is stored in its source code page. See "Character set conversions".

The file name of the external file contains the bulk load data file name, a six-digit number, and a "`.lob`" extension in the following format: *`CSVfilename_nnnnnn`*.lob. Increments start at `000001.lob`.

See also

Character set conversions on page 93

Limitations

- A bulk operation is not allowed in a manual transaction if it is not the first event.
- Once a bulk operation is started, any non-bulk operation is disallowed until the transaction is committed.

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` (enabled).
- 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 IP Address (IPAddress) option to specify the host name or the IP address of the machine where catalog tables are stored.
- Set the TCP Port (TCPPort) option to specify the port number that is assigned to the Db2 DRDA listener process on the server host machine.
- 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 Db2 Wire Protocol;Pooling=1;ConnectionReset=0;
LoadBalanceTimeout=0;MaxPoolSize=100;MinPoolSize=0;IPAddress=localhost;
TCPPort=50000;Database=Payroll;LogonID=John;Password=secret;
```

odbc.ini

```
[Db2]
Driver=ODBCHOME/lib/xxdb228.yy
...
Pooling=1
...
ConnectionReset=0
...
LoadBalanceTimeout=0
...
MaxPoolSize=100
...
MinPoolSize=0
...
IPAddress=localserver
...
TCPPort=50000
...
Database=payroll
...
LogonID=John
...
Password=secret
...
```

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.

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

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

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

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

Use Current Schema for Catalog Functions (UseCurrentSchema): If your application needs to access database objects owned only by the current user, then performance can be improved. In this case, the Use Current Schema for Catalog Functions option must be enabled. When this option is enabled, the driver returns only database objects owned by the current user when executing catalog functions. Calls to catalog functions are optimized by grouping queries. Enabling this option is equivalent to passing the Logon ID used on the connection as the SchemaName argument to the catalog functions.

Workload Manager: The Workload Manager (WLM) automatically adjusts server resources, such as CPU and memory, based on the service class associated with a Db2 workload. Therefore, an application's performance is tied to the Db2 workload to which it is assigned and, ultimately, to the service class associated with that workload. The Db2 Wire Protocol driver allows your application to set client information in the Db2 database that can be used by the WLM to classify work. If you know that your database environment uses WLM, coordinate with your database administrator to determine how setting the following options affects performance.

- **Accounting Info (AccountingInfo)**: Sets the CURRENT CLIENT_ACCTNG register (Db2 for Linux/UNIX/Windows) or the CLIENT ACCTNG register (Db2 for z/OS) on the server.
- **Application Name (ApplicationName)**: Sets the CURRENT CLIENT_APPLNAME register (Db2 for Linux/UNIX/Windows) or CLIENT APPLNAME register (Db2 for z/OS) on the server.
- **Client Host Name (ClientHostName)**: Sets the CURRENT CLIENT_WRKSTNNAME register (Db2 for Linux/UNIX/Windows) or CLIENT WRKSTNNAME register (Db2 for z/OS) on the server.
- **Client User (ClientUser)**: Sets the CURRENT CLIENT_USERID register (Db2 for Linux/UNIX/Windows) and CLIENT USERID register (Db2 for z/OS) on the server.
- **Program ID (ProgramID)**: Sets the CLIENT_PRDID value on the server.

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:

- Binding
- IBM to IANA code page values
- Cursor Stability isolation level
- Stored procedure support
- Unexpected characters
- Support for Db2 pureScale
- Persisting a result set as an XML data file
- Using arrays of parameters
- Packet logging

Binding

The driver does not work properly unless bind packages exist on every server to which you intend to connect.

IMPORTANT: You must have the appropriate privileges for the driver to create and bind packages with your user ID. These privileges are:

- BINDADD for binding packages

- CREATEIN on the collection specified by the Package Collection option
- GRANT EXECUTE on the PUBLIC group for executing the packages

These are typically the permissions of a Database Administrator (DBA). If you do not have these privileges, someone who has a user ID with DBA privileges needs to create packages by connecting with the driver.

When connecting for the first time, the driver determines whether bind packages exist on the server. If packages do not exist, the driver creates them automatically using driver data source default values.

Note: The initial driver connection to a particular server may take a few minutes because of the number and size of the packages that must be created on the server. Subsequent connections do not incur this delay.

If you change default values in a data source before connecting with the driver for the first time, the new defaults are used when creating the packages. If you want to change these values after the packages have been created, you can create or modify packages from the Modify Bindings tab of the Setup dialog. See Modify Bindings tab on page 50 for details.

On UNIX and Linux, you can also create or modify packages through a special bind utility. Depending on the platform of the Db2 server, the attribute values that must be set in the data source to bind packages are:

Db2 for Linux/UNIX/Windows, Db2 Hosted, and Db2 Warehouse on Cloud: IpAddress, Database, TcpPort

Db2 for z/OS and Db2 for i Servers: IpAddress, Location, TcpPort

Other attribute values also affect binding. See Modify Bindings tab on page 50 for details. See Connection option descriptions on page 109 for a description of these connection string attributes and their values. You must use the default values or specify new ones for these attributes in the Db2 data source section of the odbc.ini file before binding. See Configuring and connecting to data sources on page 39 for details on creating the Db2 data source.

The bind utility is located in *installation_directory*/bin. After specifying the appropriate connection string attribute values in the odbc.ini file, create or modify packages by entering the command:

```
bindxxdsn
```

where *xx* is the driver level number in the driver file name and `dsn` is the ODBC data source name in the odbc.ini file. For example:

```
bind27 Db2 Wire Protocol
```

You are prompted for a user ID and password if they are not stored in the data source. If packages are created and bound successfully, a message indicating success appears. If there are problems connecting or creating the packages, an appropriate error message appears.

Creating Db2 Packages Using List Files

You can bind the following list files on your database server to create Db2 packages:

- DDODBC_LUW.lst (Db2 for Linux/UNIX/Windows, Db2 Hosted, and Db2 Warehouse on Cloud)
- DDODBC_MVS.lst (Db2 for z/OS)
- DDODBC_400.lst (Db2 for i)

The list files are located in the \bind\LUW, \bind\zOS, and \bind\iSeries directories, respectively, in your installation directory. When you bind the list files, if any DataDirect Db2 packages exist, they will be replaced by the new packages. The list files create Db2 packages that, by default, contain 200 dynamic sections and are created in the NULLID collection. You can override the default number of dynamic sections and the collection in which the packages are created by changing the appropriate connection option in the data source, as described previously.

To create Db2 packages by binding list files:

1. Copy the appropriate list (*.lst) file and bind (*.bnd) files located in the /bind directory to a directory on the database server.
2. From the database server directory where you placed the list and bind files, start the Db2 command-line utility. Use the utility to connect to the database where you want to bind the packages. Connect using the following command:

   connect to `database_name` user `authorization_name` using `password`

   where:

   `database_name` is the name of the database to which you are connecting.

   `authorization_name` is the name of the user you are authenticating to the server.

   `password` is the user's password.

3. Execute the Db2 bind command:

   `bind @ list_file grant public`

   where *`list_file`* is the name of the list file you want to bind.

Creating Db2 Packages Manually

On Db2 for z/OS and Db2 for i servers, you can bind files manually to create Db2 packages. Refer to one of the following instruction files, as appropriate:

- CFODBC ZOS MANUAL PACKAGE CREATION INSTRUCTIONS.TXT (Db2 for z/OS)
- CFODBC AS400 MANUAL PACKAGE CREATION INSTRUCTIONS.TXT (Db2 for i)

These instruction files are located in the bind\ZOS and \bind\ISERIES directories, respectively, in your installation directory.

IBM to IANA code page values

Refer to "IBM to IANA code page values" in the *Progress DataDirect for ODBC Drivers Reference* for a list of the most commonly used IBM code pages and their IANA code page equivalents.

The IANA values are valid for the CharsetFor65535 connection string attribute and the Character Set for the CCSID 65535 option.

Cursor Stability isolation level

The Db2 Cursor Stability (CS) isolation level has been enhanced in Db2 V9.7 to reduce significantly instances of lock wait and deadlock. In previous Db2 versions, CS prevented any row that was changed by other applications from being read until the change was committed.

In this enhanced implementation, CS, where possible, avoids a read operation waiting for a row to commit before returning a value. CS now returns the currently committed result, ignoring what might happen to an uncommitted operation. Some exceptions, such as updatable cursors, exist; currently committed results cannot be returned immediately when the row might be updated based upon its previous contents. CS behavior is determined through the Concurrent Access Resolution (CAR) connection option.

Consider the following example, in which deadlocks are avoided under the currently committed semantics. In this scenario, two applications update two separate tables, but do not yet commit. Each application then attempts to read (with a read-only cursor) from the table that the other application has updated.

Table 5: Cursor Stability Examples

Step	Application A	Application B
1	`UPDATE T1 SET col1 = ? WHERE col2 = ?`	`UPDATE T2 SET col1 = ? WHERE col2 = ?`
2	`SELECT col1, col3, col4 FROM T2 WHERE col2 >= ?`	`SELECT col1, col5, FROM T1 WHERE col5 = ? AND col2 = ?`
3	`Commit`	`Commit`

Without currently committed semantics, these applications running under the cursor stability isolation level might create a deadlock, causing one of the applications to fail. This happens when each application needs to read data that is being updated by the other application.

Under currently committed semantics, if the query in step 2 (for either application) happens to require the data currently being updated by the other application, that application does not wait for the lock to be released, making a deadlock impossible. The previously committed version of the data is located and used instead.

Stored procedure support

The Db2 Wire Protocol driver supports Db2 Remote Procedure Calls (RPCs) as follows:

- Multiple result sets are returned.
- RPCs must take an argument list. The driver does not support RPCs that use a SQL descriptor area (SQLDA) data structure to specify the arguments.
- Literals are supported as stored procedure parameters.

Unexpected characters

Users are sometimes surprised when they insert a character into a database, only to have a different character displayed when they fetch it from the database. There are many reasons this can happen, but it most often involves code page issues, not driver errors.

Client and server machines in a database system each use code pages, which can be identified by a name or a number, such as Shift_JIS (Japanese) or cp1252 (Windows English). A code page is a mapping that associates a sequence of bits, called a code point, with a specific character. Code pages include the characters and symbols of one or more languages. Regardless of geographical location, a machine can be configured to use a specific code page. Most of the time, a client and database server would use similar, if not identical, code pages. For example, a client and server might use two different Japanese code pages, such as Shift_JIS and EUC_JP, but they would still share many Japanese characters in common. These characters might, however, be represented by different code points in each code page. This introduces the need to convert between code pages to maintain data integrity. In some cases, no one-to-one character correspondence exists between the two code points. This causes a substitution character to be used, which can result in displaying an unexpected character on a fetch.

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage attribute (see IANAAppCodePage on page 147). If it does not find a specific setting for IACP, it defaults to a value of ISO_8859_1 Latin_1.

If the client and server code pages are compatible, the driver transmits data in the code page of the client. Even though the pages are compatible, a one-to-one correspondence for every character may not exist. If the client and server code pages are completely dissimilar, for example, Russian and Japanese, then many substitutions occur because very few, if any, of the characters are mapped between the two code pages.

The following is a specific example of an unexpected character:

- The client machine is running the Japanese code page EUC_JP.
- The Db2 server is running the Japanese code page Shift_JIS.
- When you insert the EUC_JP code point 0xA1BD and then fetch it back, you do not see the character you expected. In fact, what displays on the client may not be a recognizable character.

This substitution occurs because the code points do not correspond in the two code pages. EUC_JP code point 0xA1BD is converted to UTF-16 code point 0x2014. Code point 0x2014 does not map to anything in Shift_JIS, resulting in the Shift_JIS substitution code point, 0x3F, being sent to, and stored in, the database. When this character is retrieved, depending on the client display, it may not display as a recognizable character.

This is not a driver error. It occurs because the code points map differently and because some characters do not exist in a code page. The best way to avoid these problems is to use the same code page on both the client and server machines.

Support for Db2 pureScale

IBM introduced Db2 pureScale to provide scaleout active and active services for IBM Db2 running on AIX on Power Systems servers. It is designed to deliver distributed availability and scalability in a clustered database system. Db2 pureScale allows a single physical Db2 database to be accessed by concurrent instances of Db2 running across several different cluster members.

A Db2 pureScale shared disk cluster is composed of a group of independent servers, or members, that cooperate as a single system. A cluster architecture such as this provides applications access to more computing power when needed, while allowing computing resources to be used for other applications when database resources are not as heavily required. For example, in the event of a sudden increase in network traffic, a Db2 pureScale cluster can distribute the load over many nodes, a feature referred to as *transaction level workload balancing*. Db2 pureScale features are available to you simply by connecting to a Db2 pureScale system with the Db2 driver. No additional configuration is required.

C*onnection failover* and *client load balancing* can be used in conjunction with a Db2 pureScale shared disk cluster, but they are not specifically part of Db2 pureScale. See Failover support on page 83 for details about how these features work.

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

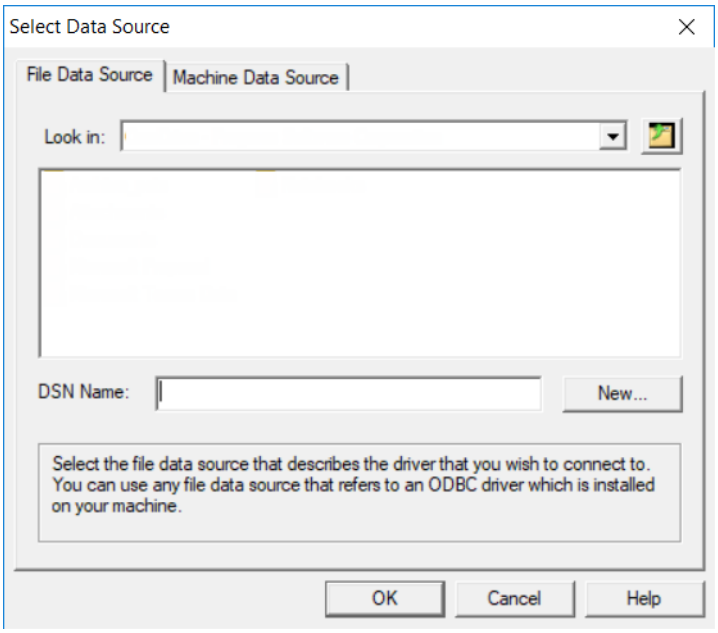
The driver for Windows is shipped with an XML persistence demo tool. This tool is installed in the product installation directory.

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.

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



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

Db2 natively supports parameter arrays and the Db2 Wire Protocol driver, in turn, supports them when connected to the Db2 databases. When designing an application for performance, using native parameter arrays for bulk inserts or updates, for example, can improve performance.

Refer to "Designing ODBC applications for performance optimization" in the *Progress DataDirect for ODBC Drivers Reference* for details.

The Db2 Wire Protocol driver accepts a proprietary statement attribute called SQL_ATTR_PARAM_ARRAY_ATOMIC. It has two values: SQL_PA_ATOMIC_YES (1) and SQL_PA_ATOMIC_NO (0).

When set to SQL_PA_ATOMIC_YES, the default, parameter array operations are atomic, meaning that if one row in the parameter array fails, then the entire array must fail.

When set to SQL_PA_ATOMIC_NO, parameter array operations are not atomic, meaning that the parameter array continues to be processed, even if one of the rows fails.

Packet logging

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

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

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

To enable TCP packet logging:

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

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

Configuring and enabling packet logging

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

Driver setup dialog (Windows)

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

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

odbc.ini file (UNIX/Linux)

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

```
[Db2 Wire Protocol]
Driver=ODBCHOME/lib/ivdb228.so
Description=DataDirect 8.0 Db2 Wire Protocol
...
Database=MyDB
...
EnablePacketLogging=1
...
IPAddress=localhost
...
LogonID=JOHN
...
PacketLoggingFilePrefix=/tmp/myPacketLog
...
PacketLoggingMaxFileSize=102400
...
PacketLoggingMaxNumFiles=10
...
Password=secret
...
TCPPort=50000
...
```

Connection string

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

```
DRIVER=DataDirect 8.0 Db2 Wire Protocol;IPAddress=localhost;TCPPort=50000;
LogonID=JOHN;Password=secret;Database=MyDB;EnablePacketLogging=1;
PacketLoggingFilePrefix=C:\temp\myPacketLog;
```

Packet logging connection options

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

Table 6: Packet Logging Connection Options

Option	Description
EnablePacketLogging	If set to 0, packet logging is disabled. This is the default. If set to 1, packet logging is enabled. If set to 2, packet logging is enabled, but the generated log file does not contain packet data. This value is typically used for performance testing. (Windows only) If set to 5, packet logging and ODBC tracing are enabled. If set to 6, packet logging and ODBC tracing are enabled, but the log file for packet logging does not contain data.
PacketLoggingFlush	If set to 0, the operating system determines when to write the log content stored in memory to disk. This is the default. If set 1, the driver determines when to write the log content stored in memory to disk. If set to 2, the content of memory is written to a the log file after each write. This setting provides a more complete logging history in the event of a crash, but can incur a performance penalty.
PacketLoggingFilePrefix	Specifies the path and prefix name of the log file. If no path is specified, the trace log resides in the working directory of the application you are using. For example: • /tmp/myLogFile (UNIX/Linux) • C:\temp\myLogFile (Windows) The above examples would generate a file named myLogFile*YYYYMMDDhhmmssxxx_nn*.out in the temp directory. If you do not specify a value for this option, the driver creates log files in the working directory using the following form: pkt*YYYYMMDDhhmmssxxx_nn*.out.
PacketLoggingMaxFileSize	Specifies the file size limit (in KB) of the log file. Once this file size limit is reached, a new log file is created and logging continues. The default is 102400. Note that subsequent files are named by appending sequential numbers, starting at 1, to the end of the original file name, for example, myLog<*timestamp*>_1.out, myLog<*timestamp*>_2.out, and so on.

Option	Description
PacketLoggingMaxNumFiles	Specifies the maximum number of log files that can be created. The default is 10. Once the maximum number of log files is created, the logging mechanism reopens the first file in the sequence, deletes the content, and continues logging in that file until the file size limit is reached, after which it repeats the process with the next file in the sequence.
PacketLoggingMemBuffSize	Specifies the maximum amount of memory, in kiloybtes, to use when writing packet logging. The default is 1024.

5

Connection option descriptions

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

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

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

Table 7: General options

Attribute (Short Name)	Default
DataSourceName (DSN)	No default value
Description (n/a)	No default value
IPAddress (IP)	No default value
TCPPort (PORT)	`50000`

User ID/password (No Encryption) options

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

Table 8: User ID/password (No Encryption) options

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`0`
Collection (COL)	No default value
Database (DB)	No default value
Location (LOC)	No default value
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 9: Kerberos authentication options

Attribute (Short Name)	Default
AuthenticationMethod (AM)	`0`
GSSClient (GSSC)	`native`

TLS/SSL encryption options

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

Table 10: TLS/SSL encryption options

Attribute (Short Name)	Default
CryptoLibName (CLN)	No default value
CryptoProtocolVersion (CPV)	`TLSv1.3,TLSv1.2`
EnableFIPS (EF)	`0` (Default provider)
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
OpenSSLConfigFile (OSSLCNF)	*`install_dir`*`\drivers\openssl.cnf` (Windows) *`install_dir`*`/lib/openssl.cnf` (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*`install_dir`*`\drivers` (Windows) *`install_dir`*`/lib` (UNIX/Linux)
SSLLibName (SLN)	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 11: Failover options

Attribute (Short Name)	Default
AlternateServers (ASRV)	No default value
ConnectionRetryCount (CRC)	`0`
ConnectionRetryDelay (CRD)	`3`
FailoverGranularity (FG)	`0` (Non-Atomic)

Attribute (Short Name)	Default
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
LoadBalancing (LB)	0 (Disabled)

Bulk load options

The following table summarizes the DataDirect Bulk Load related connection options.

Table 12: Bulk load options

Attribute (Short Name)	Default
BulkLoadBatchSize (BLBS)	1024
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadFieldDelimiter (BLFD)	No default value
BulkLoadRecordDelimiter (BLRD)	No default value
EnableBulkLoad (EBL)	0 (Disabled)

Binding options

The following table summarizes the connection options required for creating or modifying bind packages.

Table 13: Binding options

Attribute (Short Name)	Default
DynamicSections (DS)	1000
PackageCollection (PC)	NULLID
PackageOwner (PO)	No default value
GrantExecute (GE)	Enabled
GrantAuthid (GA)	PUBLIC

Timeout options

The following table summarizes timeout options.

Table 14: Timeout options

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

Pooling options

The following table summarizes pooling options.

Table 15: Pooling options

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

Client monitoring options

The following table summarizes client monitoring options.

Table 16: Client monitoring options

Attribute (Short Name)	Default
AccountingInfo (AI)	No default value
ApplicationName (AN)	No default value
ClientUser (CU)	No default value
ClientHostName (CHN)	No default value
ProgramID (PID)	No default value

Additional options

The following table summarizes additional options.

Table 17: Additional options

Attribute (Short Name)	Default
AddStringToCreateTable (ASCT)	No default value

Attribute (Short Name)	Default
AlternateID (AID)	No default value
ApplicationUsingThreads (AUT)	1 (Enabled)
CatalogSchema (CS)	No default value
CharsetFor65535 (CF6)	0
ConcurrentAccessResolution (CAR)	0 (Automatic)
CurrentFunctionPath (CFP)	No default value
DefaultIsolationLevel (DIL)	1 (READ_COMMITTED)
ExtendedOptions (XO)	No default value
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
IANAAppCodePage (IACP) LINUX ONLY	4 (ISO 8559-1 Latin-1)
KeepAlive (KA)	0 (Disabled)
MinLongVarcharSize (MINLVS)	No default value
PackageNamePrefix (PNP)	DD
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
UseCurrentSchema (UCS)	0 (Disabled)
VarcharThreshold (VT)	No default value
WithHold (WH)	1 (Enabled)
XMLDescribeType (XDT)	-10

For details, see the following topics:

- Accounting Info
- Add to Create Table
- Alternate ID
- Alternate Servers
- Application Name
- Application Using Threads
- Authentication Method

- Batch Size
- Bulk Binary Threshold
- Bulk Character Threshold
- Catalog Schema
- Character Set for CCSID 65535
- Client Host Name
- Client User
- Collection
- Concurrent Access Resolution
- Connection Pooling
- Connection Reset
- Connection Retry Count
- Connection Retry Delay
- Crypto Protocol Version
- CryptoLibName
- Current Function Path
- Data Source Name
- Database Name
- Default Isolation Level
- Description
- Dynamic Sections
- Enable Bulk Load
- Enable FIPS
- Encryption Method
- Extended Options
- Failover Granularity
- Failover Mode
- Failover Preconnect
- Fetch Time Stamp With Time Zone as Timestamp
- Field Delimiter
- Grant Execute to [check box]
- Grant Execute to [field]
- GSS Client Library

- Host Name In Certificate
- IANAAppCodePage
- IP Address
- Key Password
- Keystore
- Key Store Password
- Load Balance Timeout
- Load Balancing
- Location Name
- Login Timeout
- Max Pool Size
- Min Long Varchar Size
- Min Pool Size
- OpenSSLConfigFile
- OpenSSLProviderPath
- Package Collection
- Package Name Prefix
- Package Owner
- Password
- Program ID
- Query Timeout
- Record Delimiter
- Report Codepage Conversion Errors
- SSLLibName
- TCP Keep Alive
- TCP Port
- Trust Store
- Trust Store Password
- Use Current Schema for Catalog Functions
- User Name
- Validate Server Certificate
- Varchar Threshold
- With Hold Cursors

- XML Describe Type

Accounting Info

Attribute

AccountingInfo (AI)

Purpose

Specifies accounting information to be stored in the database. This value sets the CURRENT CLIENT_ACCTNG register (Db2 for Linux/UNIX/Windows, Db2 for i, Db2 Hosted, and Db2 Warehouse on Cloud) or the CLIENT ACCTNG register (Db2 for z/OS) in the database. This value is used by the Db2 Workload Manager.

Valid Values

string

where:

string

is the accounting information.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 95

Add to Create Table

Attribute

AddStringToCreateTable (ASCT)

Purpose

Specifies a string that is automatically added to all Create Table statements. This option is for users who need to add an In Database clause to Create Table statements.

Valid Values

string

where:

string

is valid syntax for the In Database clause of a Create Table statement.

Default

None

GUI Tab

Advanced tab

Alternate ID

Attribute

AlternateID (AID)

Purpose

Specifies the name of the default schema that is used to qualify unqualified database objects in dynamically prepared SQL statements. If the attempt to change the current schema fails, the connection fails and you receive the message `Invalid value for Alternate ID`. Refer to IBM for i documentation for permission requirements imposed by the database.

Valid Values

string

where:

string

is a valid Db2 schema name.

Default

None

GUI Tab

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

(IPAddress=*ipvalue*:TcpPort=*portvalue*:{Database | Location}=

databasevalue [, . . .])

You must specify the IP address, port number, and database name (Db2 for Linux/UNIX/Windows, Db2 Hosted, and Db2 Warehouse on Cloud) or location (Db2 for z/OS and Db2 for i) of each alternate server.

Example

The following Alternate Servers values define two alternate database servers for connection failover:

```
AlternateServers=(IpAddress=123.456.78.90:TcpPort=5177:Database=Db2DAT,
IpAddress=223.456.78.90:TcpPort=5178:Database=Db2DAT3)
```

or

```
AlternateServers=(IpAddress=123.456.78.90:
TcpPort=5177:Location=Db2DAT, IpAddress=223.456.78.90:TcpPort=5178:
Location=Db2DAT3)
```

Notes

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

Default

None

GUI Tab

Failover tab

Application Name

Attribute

ApplicationName (AN)

Purpose

Specifies the name of the application to be stored in the database. This value sets the CURRENT CLIENT_APPLNAME register (Db2 for Linux/UNIX/Windows, Db2 for i, Db2 Hosted, and Db2 Warehouse on Cloud) or CLIENT APPLNAME register (Db2 for z/OS) in the database. For Db2 for Linux/UNIX/Windows, this value also sets the APPL_NAME value of the SYSIBMADM.APPLICATIONS table. This value is used by the Db2 Workload Manager.

Valid Values

string

where:

string

is the name of the application.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 95

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 95

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.

Valid Values

`0` | `1` | `2` | `3` | `4` | `7` | `8`

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 `1` (Encrypt Password), the driver sends the user ID in clear text and an encrypted password to the server for authentication.

If set to `2` (Encrypt UID and Password), the driver sends an encrypted user ID and password to the server for authentication.

If set to `3` (Client Authentication), the driver uses client authentication when establishing a connection. The database server relies on the client to authenticate the user and does not provide additional 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 `7` (Encrypted Password AES), the driver encrypts the password with 256-bit AES encryption in the connection request. (DB2 V9.7 and higher only.)

If set to `8` (Encrypted UID and Password AES), the driver encrypts the user id and password with 256-bit AES encryption in the connection request. (DB2 V9.7 and higher only.)

Notes

- The use of AES encryption (values 7 and 8) requires that the DataDirect OpenSSL library be installed.

Default

`0` (No Encryption)

GUI Tab

Security tab

Batch Size

Attribute

BulkLoadBatchSize (BLBS)

Purpose

The number of rows that the driver sends to the database at a time during bulk operations. This value applies to all methods of bulk loading.

Valid Values

`0` | *x*

where:

x

is a positive integer that specifies the number of rows to be sent.

Default

`1024`

GUI Tab

Bulk tab

Bulk Binary Threshold

Attribute

BulkBinaryThreshold (BBT)

Purpose

The maximum size, in KB, of binary data that is exported to the bulk data file.

Valid Values

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

where:

x

is an integer that specifies the number of KB.

Behavior

If set to `-1`, all binary data, regardless of size, is written to the bulk data file, not to an external file.

If set to `0`, all binary data, regardless of size, is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file.

If set to `x`, any binary data exceeding this specified number of KB is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file.

Default

`32`

GUI Tab

Bulk tab

Bulk Character Threshold

Attribute

BulkCharacterThreshold (BCT)

Purpose

The maximum size, in KB, of character data that is exported to the bulk data file.

Valid Values

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

where:

x

is an integer that specifies the number of KB.

Behavior

If set to `-1`, all character data, regardless of size, is written to the bulk data file, not to an external file.

If set to `0`, all character data regardless of size, is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file.

If set to `x`, any character data exceeding this specified number of KB is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file.

Default

`-1`

GUI Tab

Bulk tab

Catalog Schema

Attribute

CatalogSchema (CS)

Purpose

Specifies the Db2 schema to use for Catalog functions. Specifying a schema allows you to use copies or views of the system catalog tables for catalog functions.

Valid Values

`schema_name`

where:

`schema_name`

is the name of a valid Db2 schema. If you do not specify a value for this attribute, the driver uses SYSIBM when connected to Db2 for z/OS, QSYS2 when connected to Db2 for i, and SYSCAT when connected to Linux/UNIX/Windows.

Example

Create a view `DB2ADMIN.TABLES` from `SYSCAT.TABLES`.

```
CREATE VIEW DB2ADMIN.TABLES AS SELECT * FROM SYSCAT.TABLES WHERE OWNER LIKE 'ODBC%'
```

Set CatalogSchema=DB2ADMIN, and do the SQLTables thing.

```
"TABLE_CAT", "TABLE_SCHEM", "TABLE_NAME", "TABLE_TYPE", "REMARKS"
```

The results come from the `DB2ADMIN.TABLES` view. Three rows are fetched from five columns.

```
<Null>, "DB2ADMIN", "BUG", "TABLE", <Null>
<Null>, "DB2ADMIN", "DATETEST", "TABLE", <Null>
<Null>, "DB2ADMIN", "TESTCP", "TABLE", <Null>
```

Default

None

GUI Tab

Advanced tab

Character Set for CCSID 65535

Attribute

CharsetFor65535 (CF6)

Purpose

Specifies the IANA code page to be used by the driver to convert character data stored as bit data in character columns (Char, Varchar, Longvarchar, Clob, Char for Bit Data, Varchar for Bit Data, Longvarchar for Bit Data) defined with CCSID 65535.

Valid Values

`0` | *`IANA_code_page`*

where:

`IANA_code_page`

is a valid IANA code page. Refer to "IBM to IANA code page values" in the *Progress DataDirect for ODBC Drivers Reference* for a list of the most commonly used IBM code pages and their IANA code page equivalents.

Behavior

If unspecified or set to `0`, the driver returns these columns as binary columns (SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY) and does no conversion of the data.

If an IANA code page is specified, the driver returns these columns as character columns in the character set specified. The driver does no conversion of data supplied in bound parameters.

Default

`0`

GUI Tab

Advanced tab

Client Host Name

Attribute

ClientHostName (CHN)

Purpose

Specifies the host name of the client machine to be stored in the database. This value sets the CURRENT CLIENT_WRKSTNNAME register (Db2 for Linux/UNIX/Windows, Db2 for i, Db2 Hosted, and Db2 Warehouse on Cloud) or CLIENT WRKSTNNAME register (Db2 for z/OS) in the database. This value is used by the Db2 Workload Manager.

Valid Values

`string`

where:

`string`

is the host name of the client machine.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 95

Client User

Attribute

ClientUser (CU)

Purpose

The user ID to be stored in the database. This option sets the CURRENT CLIENT_USERID register (Db2 for Linux/UNIX/Windows, Db2 for i, Db2 Hosted, and Db2 Warehouse on Cloud) and CLIENT USERID register (Db2 for z/OS) in the database. This value is used by the Db2 Workload Manager.

Valid Values

`-1` | *`client_userid`*

where:

`client_userid`

is a valid user ID.

Behavior

When set to `-1`, the driver uses the userid of the user that is currently logged onto the client.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 95

Notes

- This connection option can affect performance.

Collection

Attribute

Collection (COL)

Purpose

The current collection or library. Valid only on Db2 for z/OS and Db2 for i.

For Db2 for z/OS, this value is the user ID. If an attempt to change the current schema fails, the connection fails and you receive the message `Invalid value for Alternate ID`.

For Db2 for i, this value is the name of the schema to be used as the default qualifier for unqualified object names. If you want to access a table outside of this schema, you must specify the schema name and the table name. For example:

```
SELECT * FROM Schema.Tablename
```

Also, if the Alternate ID option is set, it overrides this option.

Valid Values

`user_ID` (Db2 for z/OS) | `schema_name` (Db2 for i)

where:

`user_ID`

is a valid user ID. Db2 permissions on the user ID must be set to SYSADM.

`schema_name`

is the default schema to use for unqualified object names.

Notes

- This option is mutually exclusive with the Database Name option.

Default

None

GUI Tab

General tab

Concurrent Access Resolution

Attribute

ConcurrentAccessResolution (CAR)

Purpose

Specifies whether a read-only query can access the currently committed value of rows that are locked by a transaction that is updating the rows. The driver must be connected to Db2 V9.7 for LUW or higher and the application isolation level must be either read committed or repeatable read.

Valid Values

`0` | `1` | `2`

Behavior

If set to `0` (Automatic), the driver persists the server behavior, as specified by the cur_commit server parameter. If cur_commit is set to "Available" or "Disable," then the current behavior, pending until the row lock is released, is used. When cur_commit is set to "On," the driver returns the last committed value of the row, regardless of whether the row is locked.

If set to `1` (Wait For Outcome), the driver always waits for the transaction to be completed before returning a row of data that has been locked by another transaction, regardless of how the cur_commit parameter is configured on the server.

If set to `2` (Use Currently Committed), the driver returns the value that was committed during the last transaction if the cur_commit parameter is configured to "On" or "Available," even though the row is locked.

Notes

- This option is ignored when connecting to a Db2 server version earlier than Db2 V9.7 for LUW.
- Db2 V10 for z/OS and Db2 for i 7.1 using DRDA do not currently support reading committed data while performing UPDATE statements. In this scenario, transactions always wait for a commit or rollback operation if they encounter data that is being updated or deleted.

Default

`0` (Automatic)

GUI Tab

Advanced tab

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.

Default

`0` (Disabled)

GUI Tab

Pooling tab

See also

Performance considerations on page 95

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 95

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

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

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

Valid Values

`0` | *x*

where:

x

is a positive integer from `1` to `65535`.

Behavior

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

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

Default

3

GUI Tab

Failover tab

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.3 | TLSv1.2

Example

If your security environment is configured to use TLSv1.3 and TLSv1.2, specify the following values:

```
CryptoProtocolVersion=TLSv1.3,TLSv1.2
```

Notes

- This option is ignored if Encryption Method is set to 0 (No Encryption) or 2 (Database Encryption).
- Consult your database administrator concerning the data encryption settings of your server.

Default

TLSv1.3,TLSv1.2

GUI Tab

Security tab

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

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

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

Valid Values

absolute_path\openssl_filename

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the CryptoLibName option. The CryptoLibName option provides a method for you to specify a cryptographic library file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.
- This option must be configured if you are using OpenSSL version 3.0.

Default

For UNIX/Linux:

Empty string

GUI Tab

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

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

See Advanced tab for details.

See also

SSLLibName on page 163

Current Function Path

Attribute

CurrentFunctionPath (CFP)

Purpose

Specifies a comma-separated list of Db2 schema names used to resolve unqualified function names and data type references in dynamically prepared SQL statements. This value also is used to resolve unqualified stored procedure names specified in CALL statements.

Valid Values

```
schema_name[,...]
```

where:

`schema_name`

is the name of a valid Db2 schema.

Default

None

GUI Tab

Advanced tab

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

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

Valid Values

`string`

where:

`string`

is the name of a data source.

Default

None

GUI Tab

General tab

Database Name

Attribute

Database (DB)

Purpose

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

Valid only for Db2 for Linux/UNIX/Windows, Db2 Hosted, and Db2 Warehouse on Cloud.

This option is mutually exclusive with the Location Name and Collection options.

Valid Values

`ext`

where:

`ext`

is the name of the one- to three-character file name extension.

This value is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option causes an error.

In other SQL statements, such as Select or Insert, users can specify an extension other than the one specified for this connection option. The Data File Extension value is used when no extension is specified.

Default

None

GUI Tab

General tab

Default Isolation Level

Attribute

DefaultIsolationLevel (DIL)

Purpose

Specifies the method by which locks on data in the database are acquired and released.

The following table shows how ODBC isolation levels map to Db2 isolation levels.

ODBC	Db2
Read Uncommitted	Uncommitted Read

ODBC	Db2
Read Committed	Cursor Stability
Repeatable Read	Read Stability
Serializable	Repeatable Read

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

Valid Values

`0` | `1` | `2` | `3` | `4`

Behavior

If set to `0` (READ_UNCOMMITTED), other processes can be read from the database. Only modified data is locked and is not released until the transaction ends.

If set to `1` (READ_COMMITTED) other processes can change a row that your application has read if the cursor is not on the row you want to change. This level prevents other processes from changing records that your application has changed until your application commits them or ends the transaction.

It also prevents your application from reading a modified record that has not been committed by another process, unless the Concurrent Access Resolution connection option is set to:

- Automatic (0) and the cur_commit server parameter is set to On
- Use Currently Committed (2) and the cur_commit server parameter is set to On or Available

In either of these cases, the application can read the last committed value. See the connection option Concurrent Access Resolution on page 128 for further details.

See Cursor Stability isolation level on page 100 for information about enhancements to the Read Committed (Cursor Stability) isolation level.

If set to `2` (REPEATABLE_READ), other processes are prevented from accessing data that your application has read or modified. All read or modified data is locked until transaction ends.

If set to `3` (SERIALIZABLE), other processes are prevented from changing records that are read or changed by your application (including phantom records) until your program commits them or ends the transaction. This level prevents the application from reading modified records that have not been committed by another process. If your application opens the same query during a single unit of work under this isolation level, the results table will be identical to the previous table; however, it can contain updates made by your application.

If set to 4 (NONE), your application can read modified records even if they have not been committed by another application. This level can only be set in the data source, not from the application. (This level is valid only on Db2 for i, and is the only isolation level that works for collections that have journaling disabled.)

Default

`1` (READ_COMMITTED)

GUI Tab

Advanced tab

Description

Attribute

Description (n/a)

Purpose

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

Valid Values

`string`

where:

`string`

is a description of a data source.

Default

None

GUI Tab

General tab

Dynamic Sections

Attribute

DynamicSections (DS)

Purpose

Specifies the maximum number of prepared statements that the driver can have open at any time.

A dynamic section is associated with a prepared statement. The driver only keeps open the number of prepared statements specified by the dynamic sections value. If the driver detects that the number of dynamic sections available in the bound Db2 packages is less than the number of dynamic sections requested in the connection string or data source, it generates the following message:

```
The current number of dynamic sections available for use is different than the number
of dynamic sections currently specified in the connection string or data source.
```

Valid Values

x

where:

x

Is a positive integer that represents a number of prepared statements.

Default

1000

GUI Tab

Modify Bindings tab

Enable Bulk Load

Attribute

EnableBulkLoad (EBL)

Purpose

Specifies the bulk load method.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver uses the database bulk load protocol when an application executes an INSERT with multiple rows of parameter data. If the protocol cannot be used, the driver returns a warning.

If set to 0 (Disabled), the driver uses standard parameter arrays.

Default

0 (Disabled)

GUI Tab

Bulk tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit. On the other platforms, the driver uses the default provider of the OpenSSL 3.5 library.
- Do not set the Truststore Password (TruststorePassword) connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be encrypted with the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates with OpenSSL 3.5-compliant algorithms" for more information.

Default

0

See also

TLS/SSL server authentication on page 73
TLS/SSL client authentication on page 78
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 80

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

The method the driver uses to encrypt data sent between the driver and the database server. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

Valid Values

0 | 1 | 2

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 set to 2 (Database Encryption), data is encrypted using the Db2 encryption protocol.

This option can only be set to 1 or 2 when Authentication Method is set to 0, 1, or 2.

Notes

- This connection option can affect performance.
- When using FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms.

Default

`0` (No Encryption)

GUI Tab

Security tab

See also

Crypto Protocol Version on page 131

Performance considerations on page 95

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).

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.

Notes

- This connection option can affect performance.

Default

0 (Connection)

GUI Tab

Failover tab

See also

Performance considerations on page 95

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 Time Stamp With Time Zone as Timestamp

Attribute

FetchTSWTZasTimestamp

Purpose

Determines whether the driver returns Timestamp with Time Zone columns as an ODBC SQL_TYPE_TIMESTAMP or as a SQL_VARCHAR data type.

Valid only for Db2 for z/OS, version 10 or higher.

Valid Values

`0` | `1`

Behavior

If set to `0` (Disabled), Timestamp with Time Zone columns are mapped to SQL_VARCHAR. Use this setting if your application needs to retrieve the information as a string.

If set to `1` (Enabled), the driver maps Timestamp with Time Zone columns to the ODBC SQL_TYPE_TIMESTAMP data type. The time zone information is truncated from the results.

Default

`0` (Disabled)

GUI Tab

Advanced tab

Field Delimiter

Attribute

BulkLoadFieldDelimiter (BLFD)

Purpose

Specifies the character that the driver will use to delimit the field entries in a bulk load data file.

Valid Values

x

where:

x

is any printable character.

For simplicity, avoid using a value that can be in the data, including all alphanumeric characters, the dash(-), the colon(:), the period (.), the forward slash (/), the space character, the single quote (') and the double quote ("). You can use some of these characters as delimiters if all of the data in the file is contained within double quotes.

Notes

- The Bulk Load Field Delimiter character must be different from the Bulk Load Record Delimiter.

Default

None

GUI Tab

Bulk tab

Grant Execute to [check box]

Attribute

GrantExecute (GE)

Purpose

Determines how EXECUTE privileges are granted on Db2 packages.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), EXECUTE privileges are granted on Db2 packages that you are creating. By default, the schema to which privileges are granted is PUBLIC.

If set to `0` (Disabled), EXECUTE privileges are granted to the schema that created the Db2 packages.

Default

`1` (Enabled)

GUI Tab

Modify Bindings tab

Grant Execute to [field]

Attribute

GrantAuthid (GA)

Purpose

Determines which Db2 schema is granted EXECUTE privileges for Db2 packages.

Valid Values

schema_name

where:

schema_name

is the name of a valid Db2 schema.

Default

PUBLIC

GUI Tab

Modify Bindings 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) when Kerberos authentication is enabled (AuthenticationMethod=4).

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 the name of the GSS client library installed on the client.

Behavior

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

If set to native, the driver uses the GSS client shipped with the operating system.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the GSS Client Library option. The GSS Client Library option provides a method for you to specify a library file used to communicate with the Key Distribution Center (KDC). However, if exposed,

the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

- The value specified for this option should be an absolute path to a mounted drive.

Default

`native`

GUI Tab

Security tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when SSL encryption is enabled (`EncryptionMethod=1`) and validation is enabled (`ValidateServerCertificate=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 *`host_name`* is specified, the driver compares the specified host name to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name with the Common Name (CN) part of the certificate. If the values do not match, the connection fails and the driver throws an exception.

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

Default

None

GUI Tab

Security tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

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

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

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

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

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

Valid Values

IANA_code_page

where:

`IANA_code_page`

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

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

IP Address

Attribute

IPAddress (IP)

Purpose

Identifies the machine where catalog tables are stored.

Valid Values

host_name | *IP_address*

where:

host_name

is the host name of the machine where catalog tables are stored. The driver must be able to find this name (with the correct address assignment) in the HOSTS file on the workstation or in a DNS server.

IP_address

is the IP address of the machine where catalog tables are stored. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.

Default

localhost

GUI Tab

General tab

Key Password

Attribute

KeyPassword (KP)

Purpose

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

Valid Values

key_password

where:

key_password

is the password of a key in the keystore.

Default

None

GUI Tab

Security tab

Keystore

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

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

Default

None

GUI Tab

Security tab

Key Store Password

Attribute

KeystorePassword (KSP)

Purpose

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

None

GUI Tab

Security tab

Load Balance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

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

Valid Values

`0` | *`x`*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

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

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

Notes

- The Min Pool Size option may cause some connections to ignore this value.

Default

`0`

GUI Tab

Pooling tab

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

Location Name

Attribute

Location (LOC)

Purpose

Specifies the name of the Db2 location that you want to access.

For Db2 for z/OS, your system administrator can determine the name of your Db2 location using the following command:

```
DISPLAY DDF
```

For Db2 for i, your system administrator can determine the name of your Db2 location using the following command. The name of the database that is listed as *LOCAL" is the value that you should use for this attribute.

```
WRKRDBDIRE
```

This option is mutually exclusive with the Database Name option.

Valid Values

`location_name`

where:

`location_name`

is the name of a valid Db2 location.

Notes

- Valid only for Db2 for z/OS and i.

Default

None

GUI Tab

General 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 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`

Notes

- This connection option can affect performance.

Example

If set to `20`, the maximum number of connections allowed in the pool is 20.

Default

`100`

GUI Tab

Pooling tab

See also

Performance considerations on page 95

Min Long Varchar Size

Attribute

MinLongVarcharSize (MINLVS)

Purpose

Specifies the minimum count of characters the driver reports for columns mapped as SQL_LONGVARCHAR. If the size of a SQL_LONGVARCHAR column is less than the value specified, the driver will increase the reported size of the column to this value when calling SQLDescribeCol and SQLColumns. This allows you to fetch SQL_LONGVARCHAR columns whose size is smaller than the minimum imposed by some third-party applications.

Valid Values

x

where:

x

is the minimum size in characters the driver will report for columns mapped to the SQL_LONGVARCHAR type.

Notes

- Configuring the VarcharThreshold and MinLongVarcharSize options allows you to fetch SQL_VARCHAR and SQL_LONGVARCHAR columns with sizes that fall between the data-type ranges used by some applications.

Default

None. If no value is specified, the driver will not change the column size reported for SQL_LONGVARCHAR columns.

GUI Tab

Advanced tab

See also

Varchar Threshold on page 169

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*

where:

x

is an integer from `1` to `65535`.

Behavior

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

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 95

OpenSSLConfigFile

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

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

Valid Values

`fips_config_file`

where:

`fips_config_file`

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

Notes

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

Default

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

OpenSSLProviderPath

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

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

Valid Values

`provider_path`

where:

`provider_path`

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

Notes

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

Default

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

Package Collection

Attribute

PackageCollection (PC)

Purpose

Specifies the name of the Db2 collection or location where the driver creates bind packages and, when required, searches for them.

Valid Values

`collection_name`

where:

`collection_name`

is a valid Db2 collection or location name.

Default

`NULLID`

GUI Tab

Modify Bindings tab

Package Name Prefix

Attribute

PackageNamePrefix (PNP)

Purpose

Specifies a two-character prefix used for package names when the driver executes dynamic SQL. The default package name uses the following syntax:

```
DDOiVRMx
```

where:

`DD`

is the two-character prefix.

`i`

is one of the following characters:

- S—Serializable : Db2 RR

- R—Repeatable Read : Db2 RS
- C—Committed Read : Db2 CS
- U—Uncommitted Read : Db2 UC
- N—Not committed: Db2 NC

VRM

is the Version Release Modification, for example, you can specify 520 to represent version 5.2.0.

x

is a one-character suffix that specifies:

- A—Cursor queries/updates
- B—Cursor queries/updates with hold
- C—Stored procedures (section 1 is for stored procedures that do not have parameters; section 2 is for procedures that do have parameters)

For example, the package name DDOC520A would represent a package using the Committed Read isolation level, at version 5.20, and using cursor queries/updates.

Valid Values

xx

where:

xx

is a two-character prefix.

Default

DD

GUI Tab

Advanced tab

Package Owner

Attribute

PackageOwner (PO)

Purpose

Specifies the AuthID assigned to the package.

Valid Values

authid

where:

authid

is a valid Db2 AuthID that has permissions to execute all the SQL in the package.

Default

None

GUI Tab

Modify Bindings Tab

Password

Attribute

Password (PWD)

Purpose

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

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Program ID

Attribute

ProgramID (PID)

Purpose

Specifies the product and version information of the driver on the client to be stored in the database. This value sets the CLIENT_PRDID value in the database. For Db2 for Linux/UNIX/Windows, this value is located in the SYSIBMADM.APPLICATIONS table. This value is used by the Db2 Workload Manager.

Valid Values

`DDTVVRRM`

where:

DDT

identifies a DataDirect Connect driver.

VV

identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version).

RR

identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release).

M

identifies a 1-character modification level (0-9 or A-Z).

Notes

- This connection option can affect performance.

Example

DDT06010

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 95

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

Specifies 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 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

Record Delimiter

Attribute

BulkLoadRecordDelimiter (BLRD)

Purpose

Specifies the character that the driver will use to delimit the record entries in a bulk load data file.

Valid Values

x

where:

x

is any printable character.

For simplicity, avoid using a value that can be in the data, including all alphanumeric characters, the dash(-), the colon(:), the period (.), the forward slash (/), the space character, the single quote (') and the double quote ("). You can use some of these characters as delimiters if all of the data in the file is contained within double quotes.

Notes

- The Bulk Load Record Delimiter character must be different from the Bulk Load Field Delimiter.

Default

None

GUI Tab

Bulk 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

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

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

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

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the SSLLibName option. The SSLLibName option provides a method for you to specify an OpenSSL library file used for SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.
- This option must be configured if you are using OpenSSL version 3.0.

Default

No default value

GUI Tab

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

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

See Advanced tab for details.

See also

CryptoLibName on page 132

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

TCP Port

Attribute

TCPPort (PORT)

Purpose

Specifies the port number that is assigned to the Db2 DRDA listener process on the server host machine.

On Db2 for i only, execute `NETSTAT` from a command line to determine the correct port number. Select option 3 to display a list of active ports on the Db2 for i machine. Find the entry for DRDA, and press F14 to toggle and display the port number. If DRDA is not currently listening, the command `CHGDDMTCPA AUTOSTART(*YES) PWDRQD(*YES)` starts the listener and ensures that it is active at IPL.

Valid Values

`IP_address` | *`service_name`*

where:

`IP_address`

is the port's IP address.

`service_name`

is the port's service name. The driver must be able to find this name (with the correct port assignment) in the SERVICES file on the workstation.

Default

`50000`

GUI Tab

General 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

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

Default

No default values

GUI Tab

Security tab

Trust Store Password

Attribute

TruststorePassword (TSP)

Purpose

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

Valid Values

`truststore_password`

where:

truststore_password

is a valid password for the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

Use Current Schema for Catalog Functions

Attribute

UseCurrentSchema (UCS)

Purpose

Specifies whether results are restricted to the tables and views in the current schema if a catalog function call is made without specifying a schema or if the schema is specified as the wildcard character %. Restricting results to the tables and views in the current schema improves performance of catalog calls that do not specify a schema.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), results of catalog function calls are restricted to the tables and views in the current schema.

If set to `0` (Disabled), results of catalog function calls are not restricted.

Default

`0` (Disabled)

GUI Tab

Advanced tab

User Name

Attribute

LogonID (UID)

Purpose

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

Valid Values

userid

where:

userid

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

Default

None

GUI Tab

Security tab

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (`EncryptionMethod=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.

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.

Notes

- Truststore information is specified using the TrustStore and TrustStorePassword options.

Default

1 (Enabled)

GUI Tab

Security tab

Varchar Threshold

Attribute

VarcharThreshold (VT)

Purpose

Specifies the threshold at which the driver describes columns of the data type SQL_VARCHAR as SQL_LONGVARCHAR. If the size of the SQL_VARCHAR column exceeds the value specified, the driver will describe the column as SQL_LONGVARCHAR when calling SQLDescribeCol and SQLColumns. This option allows you to fetch columns that would otherwise exceed the upper limit of the SQL_VARCHAR type for some third-party applications.

Valid Values

x

where:

x

is the maximum size in characters of columns the driver will describe as SQL_VARCHAR.

Notes

- Configuring the VarcharThreshold and MinLongVarcharSize options allows you to fetch SQL_VARCHAR and SQL_LONGVARCHAR columns with sizes that fall between the data-type ranges used by some applications.

Default

None. If no value is specified, the driver will not change the described type for SQL_VARCHAR columns.

GUI Tab

Advanced tab

See also

Min Long Varchar Size on page 154

With Hold Cursors

Attribute

WithHold (WH)

Purpose

Determines whether the cursor stays open on a commit.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), cursor behavior is Preserve, which keeps cursors open after a commit or rollback (SQLGetInfo( ) returns SQL_CB_PRESERVE for SQL_COMMIT_CURSOR_BEHAVIOR).

If set to `0` (Disabled), cursor behavior is Delete, which closes all cursors open after a commit or rollback (SQLGetInfo( ) returns SQL_CB_DELETE).

Default

`1` (Enabled)

GUI Tab

Advanced 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 30 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

-4

GUI Tab

Advanced tab