Progress DataDirect

Progress DataDirect for ODBC for Sybase IQ Wire Protocol Driver

User's Guide

Release 8.0.2

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for ODBC for Sybase IQ Wire Protocol Driver

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

- SAP IQ
- Sybase IQ

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:

- Additional information
- Troubleshooting
- Contacting Technical Support

What's new in this release?

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

- Release Notes
- Supported Configurations
- DatatDirect Support Matrices

Changes for 8.0.2 GA

- **Driver Enhancements**
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The 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 50 for details.
- **Changed Behavior**
  - 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

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

AIX requirements for 32-bit and 64-bit drivers

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

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 and password authentication; then, click **Apply**:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Network Address**: Type the unique identifier assigned to the Sybase IQ server machine. For example, SybaseIQserver, 2638. See Network Address on page 93 for details.
   - **Database**: Type the name of the database to which you want to connect.

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 16 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 67 provides a complete list of supported options by functionality.

- Performance considerations on page 56 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 Sybase IQ 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 44

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 and password authentication.

```
[ODBC Data Sources]
Sybase IQ=DataDirect 8.0 Sybase IQ Wire Protocol

[Sybase IQ]
Driver=ODBCHOME/lib/xxsyiq28.yy
...
Database=SybaseIQdatabase
...
NetworkAddress=SybaseIQserver, 2638
...
LogonID=jsmith
...
Password=secret
...
```

  See Configuration through the system information (odbc.ini) file on page 45 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 44, DSN-less connections, for more information. For examples, see Connection string examples on page 16.

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 16 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 67 provides a complete list of supported options by functionality.
- Performance considerations on page 56 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.

User ID and password authentication

The following string includes the options used to connect using user ID and password authentication.

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

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;NetworkAddress=network_address;
Database=database_name;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

where:

network_address

specifies the unique identifier assigned to the Sybase IQ server machine. For example, SybaseIQserver, 2638.

database_name

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

user_name

specifies your username.

password

specifies your password.

attribute=value

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

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

The following example connection string includes the options required for connecting to the database using user ID and password authentication.

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;NetworkAddress=SybaseIQserver, 2638;
Database=SybaseIQdatabase;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 44
Connection option descriptions on page 67

Connection failover

This string configures the driver to use connection failover in conjunction with some of its optional features. It uses the user ID and password 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 Sybase IQ 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. If set to `1` (Enabled), the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in a 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).

failover_mode

specifies the type of failover method the driver uses. If set to `0` (Connection), the driver provides failover protection for new connections only. If set to `1` (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress. If set to `2` (Select), the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed.

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 configuring the driver for connection failover.

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;
AlternateServers=(NetworkAddress=SybaseIQserver1, 2638:Database=SybaseIQdatabase1,
NetworkAddress=SybaseIQserver2, 2639:Database=SybaseIQdatabase2);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0;
LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 44
Connection option descriptions on page 67

Connection pooling

This string configures the driver to use 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.

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

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;Pooling=1;ConnectionReset=connection_reset;
LoadBalanceTimeout=load_balance_timeout;MaxPoolSize=max_pool_size;MinPoolSize=min_pool_size;
NetworkAddress=network_address;Database=database;LogonID=user_name;Password=password;
```

where:

connection_reset

determines whether the state of connections that are removed from the connection pool for reuse by the application is reset to the initial configuration of the connection. If set to `1`, the driver resets the state of connections and if set to `0`, the driver does not reset the state of connections.

load_balance_timeout

specifies the number of seconds to keep inactive connections open in a connection pool.

max_pool_size

specifies an integer value to specify the maximum number of connections within a single pool.

min_pool_size

specifies an integer value to specify the minimum number of connections that are opened and placed in a connection pool when it is created.

network_address

specifies the unique identifier assigned to the Sybase IQ server machine. For example, SybaseIQserver, 2638. See "Network Address" for details.

database

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

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 configuring the driver for connection pooling.

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;Pooling=1;ConnectionReset=0;
LoadBalanceTimeout=0;MaxPoolSize=100;MinPoolSize=0;NetworkAddress=SybaseIQserver1, 2638;
Database=Payroll;LogonID=John;Password=secret;
```

See also

Using a connection string on page 44
Connection option descriptions on page 67
Network Address on page 93

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 ivsyiq28.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 Sybase IQ data types are mapped to the standard ODBC data types. For Sybase IQ to Unicode data type mappings, see "Driver Specifications."

Table 1: Sybase IQ Data Type Mapping

Sybase IQ Data Type...	Maps to ODBC Data Type
BIGINT	SQL_BIGINT
BINARY	SQL_BINARY
BIT	SQL_BIT
CHAR	SQL_CHAR
DATE	SQL_TYPE_DATE
DATETIME	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
IMAGE	SQL_LONGVARBINARY
INT	SQL_INTEGER
LONG BINARY	SQL_LONGVARBINARY
LONG VARCHAR	SQL_LONGVARCHAR
MONEY	SQL_DECIMAL
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLDATETIME	SQL_TYPE_TIMESTAMP
SMALLINT	SQL_SMALLINT
SMALLMONEY	SQL_DECIMAL
TEXT	SQL_LONGVARCHAR
TIME	SQL_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP
TINYINT	SQL_TINYINT
UNIQUEIDENTIFIER	SQL_BINARY
UNIQUEIDENTIFIERSTR	SQL_CHAR
UNSIGNED BIGINT	SQL_BIGINT

Sybase IQ Data Type...	Maps to ODBC Data Type
UNSIGNED INT	SQL_INTEGER
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_VARCHAR

Note: The Sybase IQ Wire Protocol driver supports extended new limits (XNL) for character and binary columns—columns with lengths greater than 255.

See also

Driver specifications on page 23

Driver specifications

This section describes the general functionality supported by the driver.

- **ODBC compliance**: The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions.

  In addition, the following functions are supported:

  - SQLColumnPrivileges
  - SQLForeignKeys
  - SQLTablePrivileges

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

- **Unicode support**: When connected to a Unicode database, the Sybase IQ Wire Protocol driver supports the Unicode data types listed in the following table, in addition to standard ODBC data types listed in "Data Types."

  Table 2: Mapping Sybase IQ Data Types to Unicode Data Types

Sybase IQ Data Type. . .	Maps to Unicode Data Type. . .
CHAR[1]	SQL_WCHAR
LONG VARCHAR	SQL_WLONGVARCHAR
TEXT[1]	SQL_WLONGVARCHAR
UNIQUEIDENTIFIERSTR	SQL_WCHAR
VARCHAR[1]	SQL_WVARCHAR

[1] This data type is available only if the data source is configured to use the UTF-8 character set.

For data types that require the UTF-8 character set, set the Charset connection option. See "Charset" for information about using this connection option.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 applications on UNIX and Linux" for related details.

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

- **Isolation and lock levels**: The driver supports isolation levels 0 (read uncommitted), 1 (read committed, the default), 2 (repeatable read), and 3 (serializable). It supports page-level locking.

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

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

See also

Data types on page 21
Charset on page 74
UTF-16 applications on UNIX and Linux on page 34

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.

June 2022, Release 8.0.2 for the Progress DataDirect for ODBC for Sybase IQ 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 to connect to a data source and execute SQL.

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

   A command prompt opens.

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

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

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

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

Tableau (Windows only)

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

To use the driver to access data with Tableau:

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

   ```
   DataDirect Sybase IQ.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.

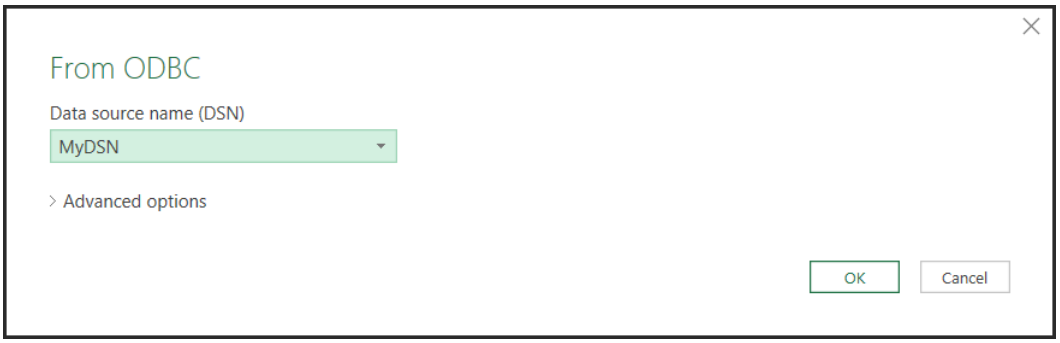
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

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.

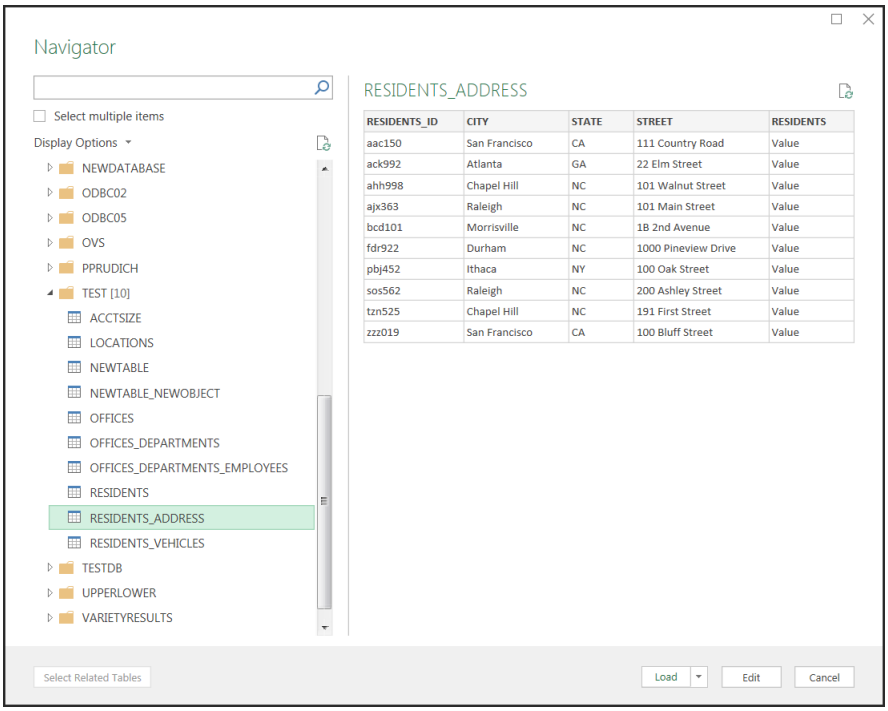
To use the driver to access data with Excel from the Data Connection Wizard:

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.

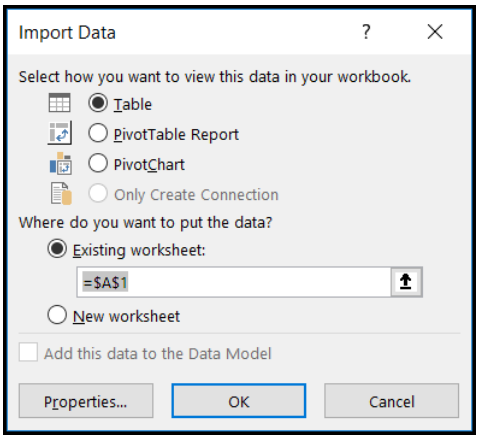


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

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
- Failover support

- DataDirect connection pooling
- Performance considerations

Environment settings

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

UNIX/Linux environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

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

- `LD_LIBRARY_PATH` on 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.

See also

Configuration through the system information (odbc.ini) file on page 45

ODBCINST

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

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

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

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

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

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

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

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

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

See also

DSN-less connections on page 48

DD_INSTALLDIR

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

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

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

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

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

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

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

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

The next step is to test load the driver.

See also

Configuration through the system information (odbc.ini) file on page 45

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/ivsyiqxx.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 Sybase IQ data source:

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

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

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

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

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

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

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

3. On the General tab, specify values for the following options:
   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Network Address**: Type the unique identifier assigned to the Sybase IQ server machine. For example, `SybaseIQserver, 2638`.
   - **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 38, provide values for options to configure advanced behavior.
6. If applicable, on the Connection tab on page 40, configure connection settings.
7. If applicalbe, on the Performance tab on page 41, configure performance settings.
8. If applicable, on the Failover tab on page 42, configure failover data source settings.
9. If applicable, on the Pooling tab on page 43, configure connection pooling settings.
10. 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.

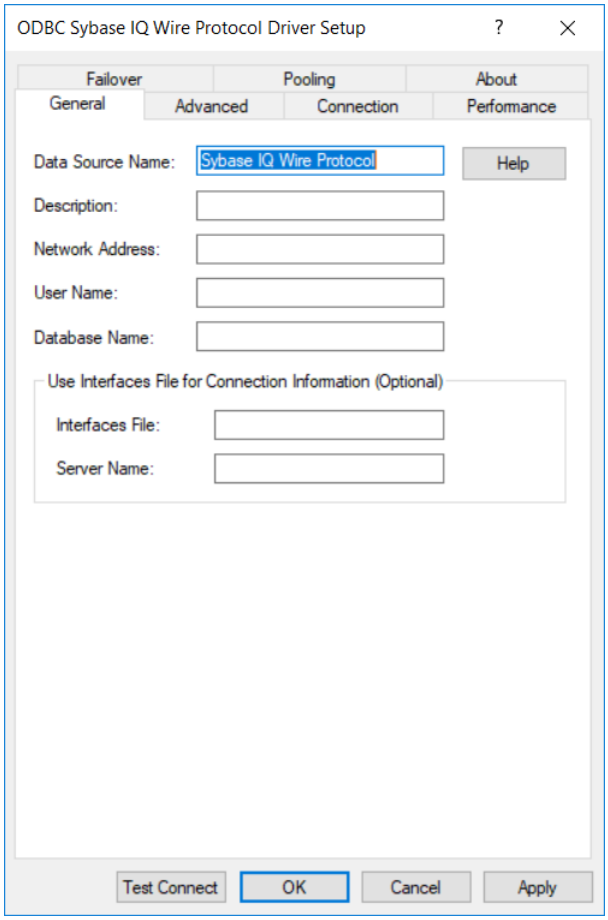
See also

Using a logon dialog box on page 51

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

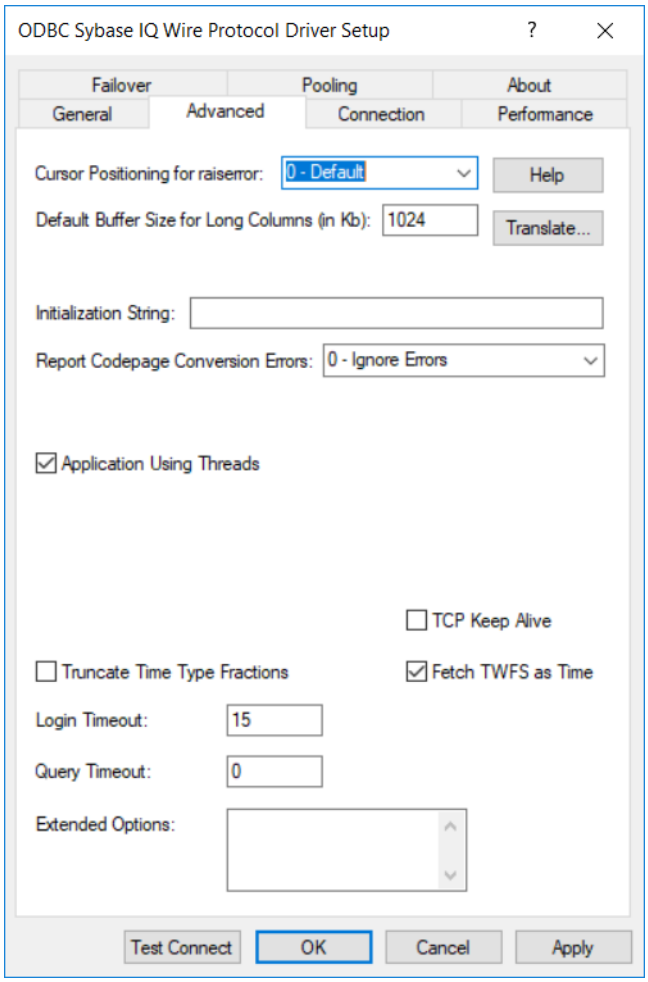


Connection Options: General	Default
Data Source Name on page 79	No default value
Description on page 80	No default value
Network Address on page 93	No default value
User Name on page 99	No default value
Interfaces File on page 88	No default value
Server Name on page 97	No default value

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
Cursor Positioning for Raiserror on page 78	**0 - Default**
Default Buffer Size for Long Columns (in KB) on page 81	**1024**
Initialization String on page 87	No default value
Report Codepage Conversion Errors on page 96	**0 - Ignore Errors**
Application Using Threads on page 73	Enabled
Truncate Time Type Fractions on page 98	Disabled

Connection Options: Advanced	Default
Fetch TWFS as Time on page 85	Enabled
Login Timeout on page 91	**15**
Query Timeout on page 95	**0**
Extended Options on page 81	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 35

Connection tab

The Connection tab allows you to specify your connection 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

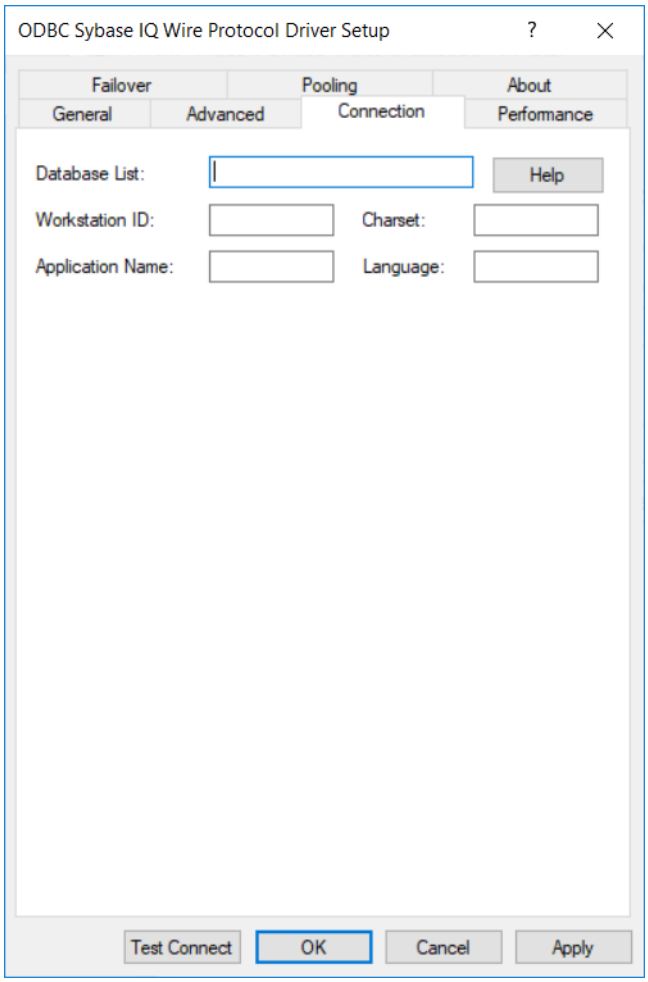


Table 3: Connection Tab Connection Options

Connection Options: Connection	Default
Database List on page 79	No default value
Workstation ID on page 99	No default value
Charset on page 74	No default value
Application Name on page 73	No default value
Language on page 89	No default value

Performance tab

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

Figure 4: Performance tab

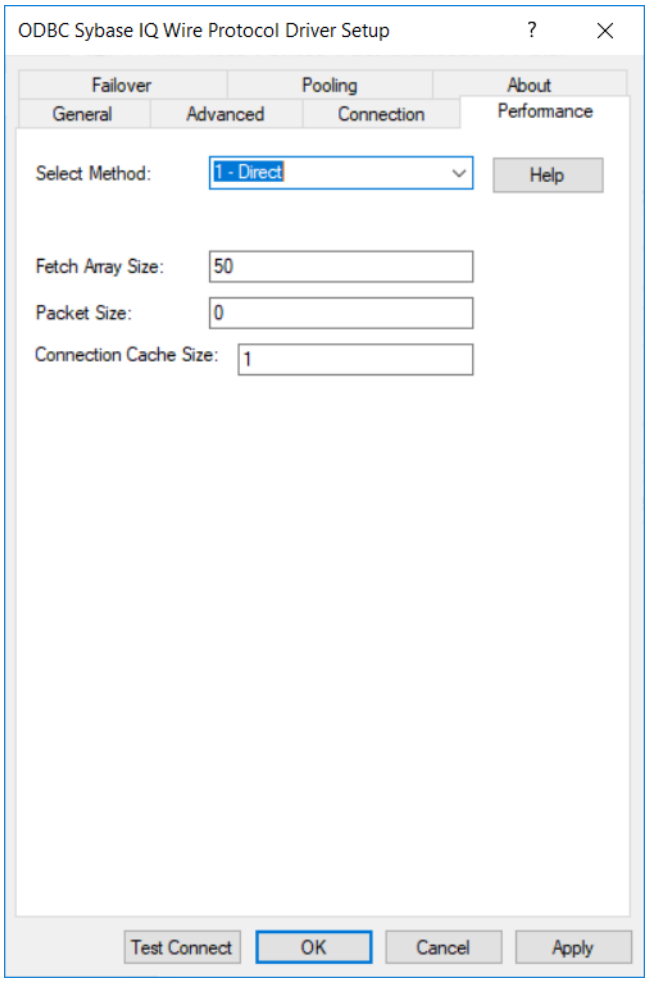


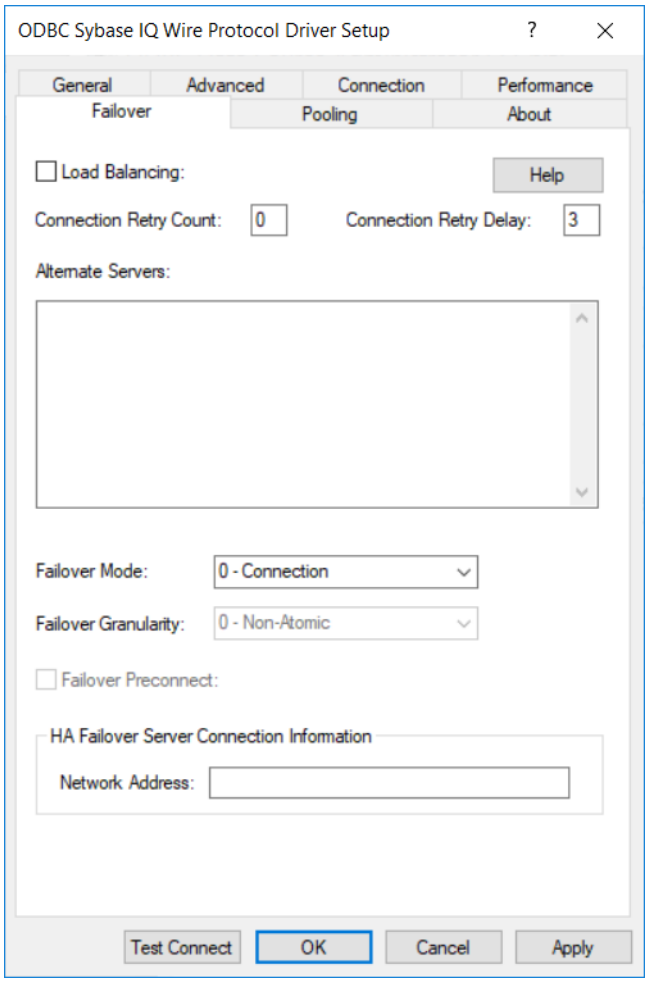
Table 4: Performance Tab Connection Options

Connection Options: Performance	Default
Select Method on page 97	**1-Direct**
Fetch Array Size on page 84	`50`
Packet Size on page 94	`0`
Connection Cache Size on page 75	`1`

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 90	Disabled
Connection Retry Count on page 77	0
Connection Retry Delay on page 77	3
Alternate Servers on page 72	No default value
Failover Mode on page 83	**0 - Connection**
Failover Granularity on page 82	**0 - Non-Atomic**

Connection Options: Failover	Default
Failover Preconnect on page 84	Disabled
Network Address on page 93	No default value

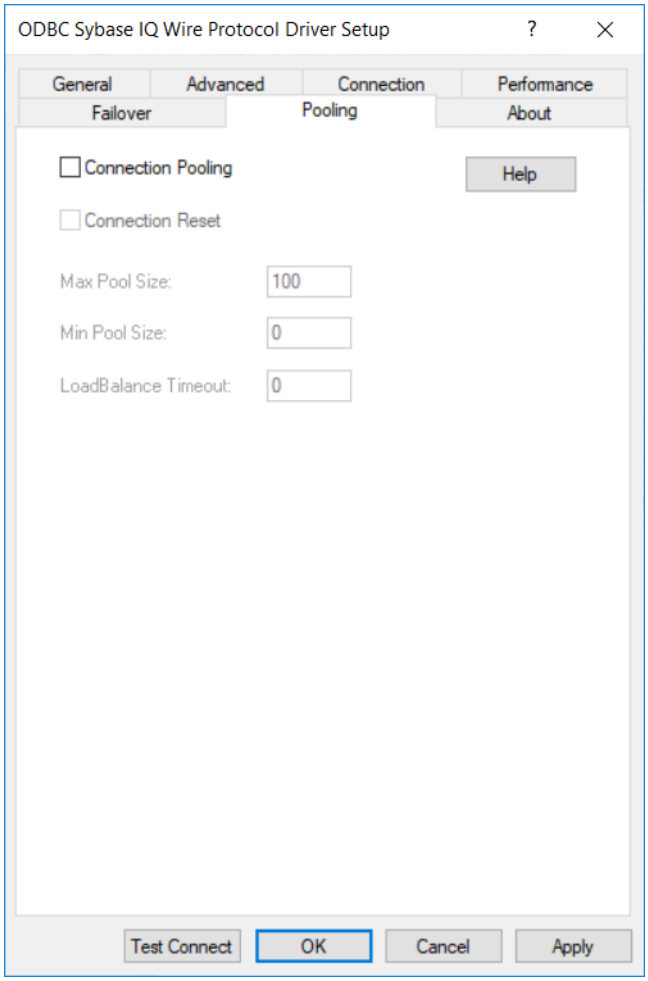
See also

Configuring the driver using the GUI on page 35

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 75	Disabled
Connection Reset on page 76	Disabled

Connection Options: Pooling	Default
Max Pool Size on page 92	100
Min Pool Size on page 92	0
Load Balance Timeout on page 89	0

See also

Configuring the driver using the GUI on page 35

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=SYBIQTABLES;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

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

```
FILEDSN=SYBIQ.dsn;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER={DataDirect 8.0 Sybase IQ Wire Protocol};NA=123.456.78.90,2638;
DB=SYBIQACCT;UID=JOHN;PWD=XYZZY
```

See also

Connection option descriptions on page 67

Connection string examples on page 16

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:

```
Sybase IQ=DataDirect 8.0 Sybase IQ Wire Protocol Driver
```

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

The `[ODBC Data Sources]` section also includes data source definitions. The default `odbc.ini` contains a data source definition for the driver. Each data source definition begins with a data source name in square brackets, for example, `[Sybase IQ]`. 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 67
Sample default odbc.ini file on page 46
File data sources on page 49
IANAAppCodePage on page 87
DSN-less connections on page 48

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]
Sybase IQ Wire Protocol=DataDirect 8.0 Sybase IQ Wire Protocol

[Sybase IQ Wire Protocol]
Driver=ODBCHOME/lib/ivsyiq28.so
Description=DataDirect 8.0 Sybase IQ Wire Protocol
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=50
Charset=
```

```
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CursorCacheSize=1
Database=<database_name>
DefaultLongDataBuffLen=1024
FailoverGranularity=0
FailoverMode=0
FailoverNetworkAddress=
FailoverPreconnect=0
FetchTWFSasTime=1
InitializationString=
InterfacesFile=
InterfacesFileServerName=
Language=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
NetworkAddress=<SybaseIQ_host,SybaseIQ_server_port>
OptimizePrepare=1
PacketSize=0
Password=
Pooling=0
QueryTimeout=0
RaiserrorPositionBehavior=0
ReportCodePageConversionErrors=0
SelectMethod=0
TruncateTimeTypeFractions=0
WorkStationID=

[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 67

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 Sybase IQ 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 33
Configuration through the system information (odbc.ini) file on page 45

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 Sybase IQ Wire Protocol=Installed

[DataDirect 8.0 Sybase IQ Wire Protocol]
Driver=ODBCHOME/lib/ivsyiq28.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 Sybase IQ Wire Protocol
...
NetworkAddress=123.456.78.90, 2638
...
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\SybaseIQ.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/SybaseIQ.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/SybaseIQ.dsn;DB=payroll;UID=Jsmith;PWD=secret
```

See also

Configuring and connecting to data sources on page 31
DSN-less connections on page 48
Additional configuration methods for UNIX and Linux on page 45

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`*

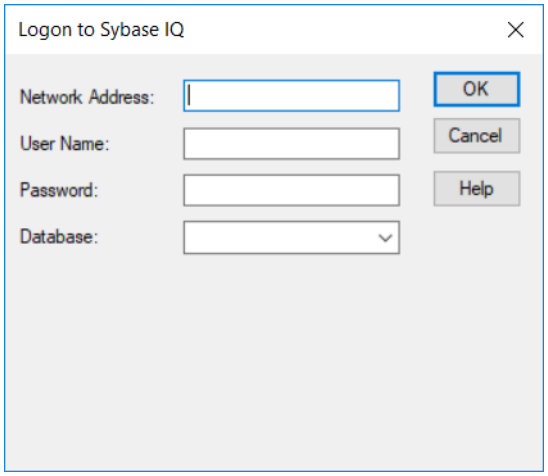
4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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 7: Logon to Sybase IQ dialog box



In this dialog box, provide the following information:

1. In the Network Address field, type the unique identifier assigned to the Sybase IQ server machine. For example, SybaseIQserver, 2638.
2. In the User Name field, type your Sybase IQ user name.
3. In the Password field, type your password.
4. In the Database field, type the name of the database to which you want to connect
5. Click **OK** to complete the logon.

Authentication

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

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

- Set the Network Address (NetworkAddress) option to specify the unique identifier assigned to the Sybase IQ server 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 the connection information required to establish a connection using user ID and password authentication.

Connection string

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;NetworkAddress=123.456.78.90, 2638;
Database=payroll;LogonID=jsmith;Password=secret;
```

odbc.ini

```
[Sybase IQ]
Driver=ODBCHOME/lib/xxsyiq28.yy
...
NetworkAddress=123.456.78.90, 2638
...
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.

See also

Connection option descriptions on page 67

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.

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. These examples use the user ID and password authentication method for authentication.

Connection string

```
DRIVER=DataDirect 8.0 Sybase IQ Wire Protocol;
AlternateServers=(NetworkAddress=SybaseIQserver1, 2638:Database=payroll,
NetworkAddress=SybaseIQserver2, 2638:Database=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/ivsyiqxx.so
Description=DataDirect Sybase IQ Wire Protocol
...
AlternateServers=(NetworkAddress=SybaseIQserver1,
2638:Database=payroll,NetworkAddress=SybaseIQserver2, 2638:Database=accounting)
...
ConnectionRetryCount=4
...
ConnectionRetryDelay=5
...
LoadBalancing=0
...
FailoverMode=1
...
FailoverPreconnect=1
...
LogonID=John;
...
Password=secret;
...
```

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

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

See also

Connection option descriptions on page 67

Guidelines for primary and alternate servers

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

DataDirect connection pooling

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

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

To configure the driver to use connection pooling:

- Set the Connection Pooling (Pooling) option to `1` (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 Network Address (NetworkAddress) option to specify the unique identifier assigned to the Sybase IQ server 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 Sybase IQ Wire Protocol;Pooling=1;ConnectionReset=0;
LoadBalanceTimeout=0;MaxPoolSize=100;MinPoolSize=0;NetworkAddress=SybaseIQserver1, 2638;
Database=Payroll;LogonID=John;Password=secret;
```

odbc.ini

```
[Sybase IQ]
Driver=ODBCHOME/lib/xxsyiq28.yy
...
Pooling=1
...
ConnectionReset=0
...
LoadBalanceTimeout=0
...
MaxPoolSize=100
...
MinPoolSize=0
...
NetworkAddress=SybaseIQserver1, 2638
...
Database=payroll
...
LogonID=John
...
Password=secret
...
```

See also

Connection option descriptions on page 67

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.

Default Buffer Size for Long/LOB Columns (DefaultLongDataBuffLen): To improve performance when your application fetches images, pictures, or long text or binary data, a buffer size can be set to accommodate the maximum size of the data. The buffer size should only be large enough to accommodate the maximum amount of data retrieved; otherwise, performance is reduced by transferring large amounts of data into an oversized buffer. If your application retrieves more than 1 MB of data, the buffer size should be increased accordingly.

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.

Fetch Array Size (ArraySize): If the Select Method connection option is set to 0 and your application fetches more than 50 rows at a time, you should set Fetch Array Size to the approximate number of rows being fetched. This reduces the number of round trips on the network, thereby increasing performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network. You should use Fetch Array Size in conjunction with Select Method.

Note: The ideal setting for your application will vary. To calculate the ideal setting for this option, you must know the size in bytes of the rows that you are fetching and the size in bytes of your Network Packet. Then, you must calculate the number of rows that will fit in your Network Packet, leaving space for packet overhead. For example, suppose your Network Packet size is 1024 bytes and the row size is 8 bytes. Dividing 1024 by 8 equals 128; however, the ideal setting for Fetch Array Size is 127, not 128, because the number of rows times the row size must be slightly smaller than the Network Packet size.

Packet Size (PacketSize): Typically, it is optimal for the client to use the maximum packet size that the database server allows. This reduces the total number of round trips required to return data to the client, thus improving performance. Therefore, performance can be improved if the PacketSize attribute is set to the maximum packet size of the server.

Select Method (SelectMethod): If your application often executes a SQL statement before processing or closing the previous result set, then it uses multiple active statements per connection. An active statement is defined as a statement where all the result rows or result sets have not been fetched. Using multiple active statements can cause high overhead on the server. The default setting (1) of this option causes the driver to execute statements directly without the use of database cursors and limits the application to one active statement per connection. If your application requires multiple active statements, then set Select Method to 0 (Cursor). Keep in mind that you may see a negative impact in performance. If this option is set to 0, it should be used in conjunction with Fetch Array Size (ArraySize). If this option is set to 1, Fetch Array Size (ArraySize) has no effect.

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:

- NULL values
- Persisting a result set as an XML data file
- Unexpected characters
- Using arrays of parameters
- Packet logging

NULL values

When the Sybase IQ Wire Protocol driver establishes a connection, the driver sets the Sybase database option ansinull to on. Setting ansinull to on ensures that the driver is compliant with the ANSI SQL standard, which makes developing cross-database applications easier.

By default, Sybase IQ does not evaluate NULL values in SQL equality (=), inequity (<>), or aggregate function comparisons in an ANSI SQL-compliant manner. For example, the ANSI SQL specification defines that `col1=NULL` always evaluates to false:

```
SELECT * FROM table WHERE col1 = NULL
```

Using the default database setting (ansinull=off), the same comparison evaluates to true instead of false.

Setting ansinull to on changes the default database behavior so that SQL statements must use `IS NULL` instead of `=NULL`. For example, using the Sybase IQ Wire Protocol driver, if the value of col1 in the following statement is NULL, the comparison evaluates to true:

```
SELECT * FROM table WHERE col1 IS NULL
```

In your application, you can restore the default Sybase IQ behavior for a connection in the following ways:

- Use the Initialization String option to specify the SQL command `set ANSINULL off`. For example, the following connection string ensures that the handling of NULL values is restored to the Sybase IQ default for the current connection:

  ```
  DSN=SYB TABLES;DB=PAYROLL;IS=set ANSINULL off
  ```

- Explicitly execute the following statement after the connection is established:

  ```
  SET ANSINULL OFF
  ```

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.

Argument	Definition
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

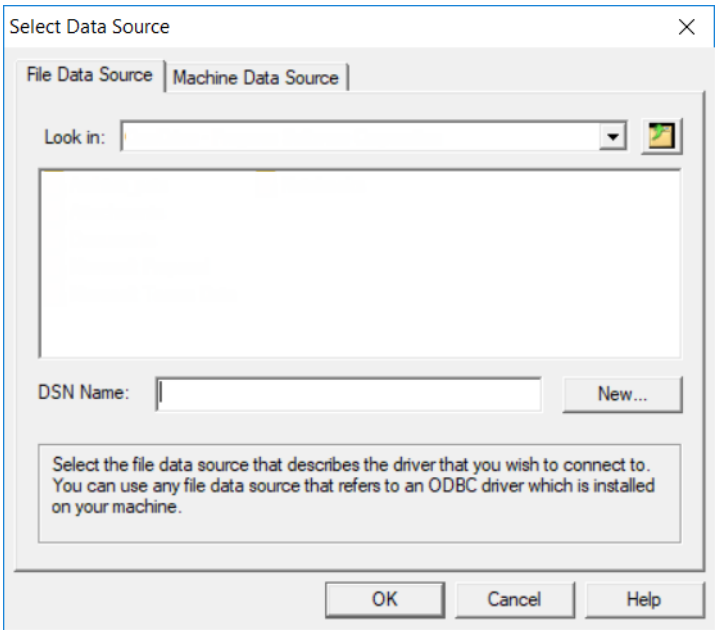
The driver for Windows is shipped with an XML persistence demo tool. This tool is installed in the product installation directory.

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.

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 and server code pages are compatible, the driver transmits data in the code page of the server. 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 Windows client machine is running code page cp1252.
- The Sybase IQ server is running code page cp850.
- You insert decimal literals for character data. You think you are inserting LATIN SMALL LETTER I WITH ACUTE (í) and BOX DRAWINGS DOUBLE VERTICAL (||) in the database. When you fetch the data, you see INVERTED EXCLAMATION MARK (¡) and MASCULINE ORDINAL INDICATOR (º) displayed on the client instead.

This occurs because the code points do not correspond in the two code pages. An example of syntax you would use to insert the decimal literals is:

```
CREATE table cp850chars(val text)
INSERT INTO cp850chars values(CHAR(161)+CHAR(186))
```

This effectively inserts the hexadecimal bytes for the numbers 161 (0xA1) and 186 (0xBA) into the text column. Each of these hexadecimal bytes is treated as the single byte code point for the character it represents. The problem is that the character representation for these two particular hexadecimal values is different from code page cp850 to code page cp1252. On cp850, these hexadecimal values represent í (0xA1) and || (0xBA), which is what you thought you were inserting by using the previously described syntax. When you fetch these hexadecimal values, however, the characters displayed on your client machine are ¡ (0xA1) and º (0xBA), because that is what the hexadecimal values represent in code page cp1252. This is not a matter of data corruption or substitution; these hexadecimal values simply represent different values in the two different code pages.

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.

Using arrays of parameters

When designing an application, using 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.

Because Sybase IQ databases do not support parameter arrays natively, the Sybase IQ Wire Protocol driver emulates them by sending T-SQL batches of Insert or Update statements to the database, which will improve performance.

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.

```
[Sybase IQ Wire Protocol]
Driver=ODBCHOME/lib/ivsyiq28.so
Description=DataDirect 8.0 Sybase IQ Wire Protocol
...
Database=MyDB
...
EnablePacketLogging=1
...
LogonID=JOHN
...
NetworkAddress=SybaseIQserver, 2638
...
PacketLoggingFilePrefix=/tmp/myPacketLog
...
PacketLoggingMaxFileSize=102400
...
PacketLoggingMaxNumFiles=10
...
Password=secret
...
```

Connection string

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

```
DRIVER={DataDirect 8.0 Sybase IQ Wire Protocol};
NetworkAddress=SybaseIQserver, 2638;Database=MyDB;
LogonID=JOHN;Password=secret;EnablePacketLogging=1;
PacketLoggingFilePrefix=C:\temp\myPacketLog;
```

Packet logging connection options

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

Table 5: Packet Logging Connection Options

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

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

5

Connection option descriptions

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

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 6: General options

Attribute (Short Name)	Default
DataSourceName (DSN)	No default value
Database (DB)	No default value
Description (n/a)	No default value
InterfacesFile (IF)	No default value
InterfacesFileServerName (IFSN)	No default value
NetworkAddress (NA)	No default value

User ID and password options

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

Table 7: User ID and password options

Attribute (Short Name)	Default
LoginID (UID)	No default value
Password (PWD)	No default value

Failover options

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

Table 8: Failover options

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

Timeout options

The following table summarizes timeout options.

Table 9: Timeout options

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

Pooling options

The following table summarizes connection pooling options.

Table 10: Pooling options

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

Connection options

The following table summarizes the options provided on the Connection tab of the setup dialog box.

Table 11: Connection options

Attribute (Short Name)	Default
ApplicationName (APP)	No default value
Charset (CS)	No default value
DatabaseList (n/a)	No default value
Language (LANG)	English
WorkstationID (WKID)	No default value

Performance options

The following table summarizes the options that can affect performance.

Table 12: Performance options

Attribute (Short Name)	Default
ArraySize (AS)	50
ConnectionCacheSize (CCS)	1
PacketSize (PS)	0
SelectMethod (SM)	1 (Direct)

Additional options

The following table summarizes additional options.

Table 13: Additional options

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
DefaultLongDataBuffLen (DLDBL)	1024
ExtendedOptions (XO)	No default value
FailoverNetworkAddress (FNA)	No default value
FetchTWFSasTime (FTWFSAT)	1 (Enabled)
IANAAppCodePage (IACP) LINUX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	No default value
InterfacesFile (IF)	No default value
RaiserrorPositionBehavior (REPB)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
TruncateTimeTypeFractions (TTF)	0 (Disabled)

For details, see the following topics:

- Alternate Servers
- Application Name
- Application Using Threads
- Charset
- Connection Cache Size
- Connection Pooling

- Connection Reset
- Connection Retry Count
- Connection Retry Delay
- Cursor Positioning for Raiserror
- Data Source Name
- Database List
- Database Name
- Description
- Default Buffer Size for Long Columns (in KB)
- Extended Options
- Failover Granularity
- Failover Mode
- Failover Preconnect
- Fetch Array Size
- Fetch TWFS as Time
- HA Failover Server Connection Information/Network Address
- IANAAppCodePage
- Initialization String
- Interfaces File
- Language
- Load Balance Timeout
- Load Balancing
- Login Timeout
- Max Pool Size
- Min Pool Size
- Network Address
- Packet Size
- Password
- Query Timeout
- Report Codepage Conversion Errors
- Select Method
- Server Name
- Truncate Time Type Fractions

- User Name
- Workstation ID

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

```
({NetworkAddress=addressvalue | InterfacesFileServerName=sectionvalue}[, ...])
```

NetworkAddress and InterfacesFileServerName can be used in the same string.

Notes

- An alternate server address in IPv6 format must be enclosed in double quotation marks.
- You must specify the network address of each alternate database server or the section in the Interfaces file that contains the network connection information for the Sybase IQ database server you want to access (InterfacesFileServerName).
- The Alternate Servers option and the HA Failover Server Connection Information option are mutually exclusive.

Example

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

```
(InterfacesFileServerName=Accounting, NetworkAddress="255.125.1.11, 4200",
NetworkAddress="SybaseIQ2, 4200")
```

In this example, the network address of the last two alternates contain commas. In this case, enclose the network address with double quotation marks as shown.

Default

None

GUI Tab

Failover tab

Application Name

Attribute

ApplicationName (APP)

Purpose

The name used by Sybase IQ to identify your application.

Valid Values

string

where:

string

is a valid application name.

Default

None

GUI Tab

Connection tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

- This connection option can affect performance.

Default

1 (Enabled)

GUI Tab

Advanced tab

See Also

Performance considerations on page 56

Charset

Attribute

Charset (CS)

Purpose

The name of a character set installed on the Sybase IQ server to be used by the driver.

This option is not a substitute for the IANAAppCodePage option. See "IANAAppCodePage" for details.

Valid Values

charset

where:

charset

is the name of a character set installed on the Sybase IQ server.

Behavior

If unspecified, the character set setting on the Sybase IQ server is used.

For the driver to return Unicode SQL types, use a value of UTF-8. Refer to the Sybase IQ server documentation for a list of valid character sets.

Example

If your client needs to receive data in iso-8859-1 from a non-Unicode Sybase IQ server, you would specify a value of `iso_1`.

Default

No default value

GUI Tab

Connection tab

See Also

IANAAppCodePage on page 87

Connection Cache Size

Attribute

ConnectionCacheSize (CCS)

Purpose

The number of connections that the connection cache can hold.

Valid Values

x

where:

x

is a positive integer representing the number of connections that the connection cache can hold.

To enable the connection cache, you must set the Select Method option to `1` (Direct). Increasing the connection cache may increase performance of some applications but requires additional database resources.

Default

`1`

GUI Tab

Performance tab

See Also

Select Method on page 97

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

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Pooling tab

See Also

Performance considerations on page 56

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 56

Connection Retry Count

Attribute

ConnectionRetryCount (CRC)

Purpose

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

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

Valid Values

`0` | *x*

where:

x

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

Behavior

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

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

Default

`0`

GUI Tab

Failover tab

Connection Retry Delay

Attribute

ConnectionRetryDelay (CRD)

Purpose

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

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

Valid Values

0 | *x*

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

3

GUI Tab

Failover tab

Cursor Positioning for Raiserror

Attribute

RaiserrorPositionBehavior (REPB)

Purpose

Determines whether the driver returns raiserrors when the next statement is executed or handles them separately.

Valid Values

0 | 1

Behavior

If set to 0 (Default), raiserrors are handled separately from surrounding statements. The error is returned when a raiserror is processed (for example, resulting from SQLExecute, SQLExecDirect, or SQLMoreResults). The result set is empty.

If set to 1 (Microsoft compatible), raiserrors are returned when the next statement is processed, and the cursor is positioned on the first row of the subsequent result set. This could result in multiple raiserrors being returned on a single execute.

Default

0 (Default)

GUI Tab

Advanced tab

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

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

Valid Values

string

where:

string

is the name of a data source.

Default

No default value

GUI Tab

General tab

Database List

Attribute

n/a

Description

A list of database names that will appear in the drop-down list of the logon dialog box. See "Using a logon dialog box" for a description.

Valid Values

database_list

where:

database_list

is a comma-separated list of database names that will appear in the drop-down list of the logon dialog box.

Default

No default value

GUI Tab

Connection tab

Database Name

Attribute

Database (DB)

Purpose

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

Valid Values

database_name

where:

database_name

is the name of a valid database. If you do not specify a value, the default is the database defined by the system administrator for each user.

Default

No default value

GUI Tab

General tab

Description

Attribute

Description (n/a)

Purpose

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

Valid Values

string

where:

string

is a description of a data source.

Default

No default value

GUI Tab

General tab

Default Buffer Size for Long Columns (in KB)

Attribute

DefaultLongDataBuffLen (DLDBL)

Purpose

The maximum length of data (in KB) the driver can fetch from long columns in a single round trip and the maximum length of data that the driver can send using the SQL_DATA_AT_EXEC parameter.

This option also applies to binding long parameters in chunks. The driver truncates any data passed in a Long/LOB SQL_DATA_AT_EXEC parameter to the size specified.

Valid Values

An integer in multiples of 1024

The value must be in multiples of 1024 (for example, 1024, 2048). You need to increase the default value if the total size of any Long data exceeds 1 MB. This value is multiplied by 1024 to determine the total maximum length of fetched data. For example, if you enter a value of 2048, the maximum length of data would be 1024 x 2048, or 2097152 (2 MB).

Notes

- This connection option can affect performance.

Default

`1024`

GUI Tab

Advanced tab

See Also

Performance considerations on page 56

Extended Options

Attribute

ExtendedOptions (xo)

Purpose

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

Valid Values

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

where:

option

is a connection option.

value

is the value or setting of the connection option.

Default Value

No default value

Failover Granularity

Attribute

FailoverGranularity (FG)

Purpose

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

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

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

Valid Values

0 | 1 | 2 | 3

Behavior

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

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

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

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

Default

0 (Non-Atomic)

GUI Tab

Failover tab

Failover Mode

Attribute

FailoverMode (FM)

Purpose

Specifies the type of failover method the driver uses.

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

Notes

- This connection option can affect performance.

Default

0 (Connection)

GUI Tab

Failover tab

See Also

Performance considerations on page 56

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 Array Size

Attribute

ArraySize (AS)

Purpose

The number of rows the driver retrieves from the server for a fetch. This is not the number of rows given to the user. You should use Fetch Array Size in conjunction with Select Method.

Valid Values

x

where:

x

is a positive integer specifying the number of rows.

Notes

- This connection option can affect performance.

Default

`50`

GUI Tab

Performance tab

See Also

Select Method on page 97

Performance considerations on page 56

Fetch TWFS as Time

Attribute

FetchTWFSasTime (FTWFSAT)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

`1` (Enabled)

GUI Tab

Advanced tab

HA Failover Server Connection Information/Network Address

Attribute

FailoverNetworkAddress (FNA)

Purpose

The network address of the High Availability (HA) Failover server to be used in the event of a connection loss. The driver detects the dropped connection and automatically reconnects to the specified HA Failover server. This option is valid only for Sybase IQ servers that have the High Availability Failover feature enabled.

Valid Values

`IP_`*`address`*`,` *`port_number`* | *`server_name`*`,` *`port_number`*

where:

`IP_ address`

is the IP address that uniquely identifies the HA Failover server.

`port_number`

is the port number assigned to the listener process on the HA Failover server.

`server_name`

is a name that uniquely identifies the HA Failover server. You can use this format if your environment supports named servers.

Notes

- The HA Failover Server Connection Information option and the Alternate Servers option are mutually exclusive.

Example

```
199.226.224.34, 2638
```

```
Sybaseiqserver, 2638
```

Default

No default value

GUI Tab

Failover tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled or if your database character set is not Unicode. The value you specify must match the database character encoding and the system locale.

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

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

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

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

Valid Values

IANA_code_page

where:

IANA_code_page

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

Notes

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

Default

```
4 (ISO 8559-1 Latin-1)
```

GUI Tab

N/A

Initialization String

Attribute

InitializationString (IS)

Purpose

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

Valid Values

SQL_command

where:

SQL_command

is a valid SQL command that is supported by the database.

Notes

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

Example

To allow delimited identifiers, specify:

```
Initialization String=set QUOTED_IDENTIFIER on
```

Default

No default value

GUI Tab

Advanced tab

Interfaces File

Attribute

InterfacesFile (IF)

Purpose

The directory to the Interfaces file.

Valid Values

file_dir

where:

file_dir

is the directory to the Interfaces file.

Behavior

If unspecified and a value is specified for the Server Name option, the driver looks for the path name of the Interfaces file in the Registry under HKEY_LOCAL_MACHINE\SOFTWARE\DataDirect\InterfacesFile. If this Registry value is empty, the driver will try to open the SQL.INI file found in the same directory where the driver is located and use it as the Interfaces file.

Notes

- This option and the Network Address option are mutually exclusive.

Default

No default value

GUI Tab

General tab

Language

Attribute

Language (LANG)

Purpose

The national character set installed on the Sybase IQ server.

Valid Values

charset

where:

charset

is the national character set installed on the Sybase IQ server.

Default

None (English)

GUI Tab

Connection tab

Load Balance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

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

Valid Values

`0` | *x*

where

x

is a positive integer that specifies a number of seconds.

Behavior

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

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

Notes

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

This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Pooling tab

See Also

Performance considerations on page 56

Load Balancing

Attribute

LoadBalancing (LB)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

`0` (Disabled)

GUI Tab

Failover tab

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

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

Valid Values

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

where:

x

is a positive integer that represents a number of seconds.

Behavior

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

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

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

Default

`15`

GUI Tab

Advanced tab

Max Pool Size

Attribute

MaxPoolSize (MXPS)

Purpose

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

Valid Values

An integer from `1` to `65535`

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

Notes

- This connection option can affect performance.

Default

`100`

GUI Tab

Pooling tab

See Also

Performance considerations on page 56

Min Pool Size

Attribute

MinPoolSize (MNPS)

Purpose

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

Valid Values

`0` | *x*

Behavior

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

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

Notes

- This connection option can affect performance.

Example

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

Default

`0`

GUI Tab

Pooling tab

See Also

Performance considerations on page 56

Network Address

Attribute

NetworkAddress (NA)

Purpose

A unique identifier assigned to the Sybase IQ server machine.

Valid Values

`server_name` | *`IP_address`*

where:

`server_name`

is the Sybase IQ server name specified as: *`named_server`*, *`port_number`*. For example, you can enter `SybaseIQserver, 2638`.

`IP_address`

is the Sybase IQ server address specified as: *`IP_address`*, *`port_number`*. For example, you can enter `199.226.224.34, 2638`. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.

Notes

- This option is mutually exclusive with the Interfaces File and the Server Name option.

Default

No default value

GUI Tab

General tab

Packet Size

Attribute

PacketSize (PS)

Purpose

Determines the number of bytes for each database protocol packet that is transferred from the database server to the client machine. Adjusting the packet size can improve performance. The optimal value depends on the typical size of data that is inserted, updated, or returned by the application and the environment in which it is running. Typically, larger packet sizes work better for large amounts of data. For example, if an application regularly returns character values that are 10,000 characters in length, using a value of 32 (16 KB) typically results in improved performance.

Valid Values

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

Behavior

If set to `-1`, the driver uses the maximum packet size that is set by the database server.

If set to `0`, the driver uses the default packet size that is used by the database server.

If set to *x*, an integer from 1 to 127, the driver uses a packet size that is a multiple of 512 bytes. For example, PacketSize=8 means to set the packet size to 8 * 512 bytes (4096 bytes).

Notes

- The ODBC connection attribute SQL_ATTR_PACKET_SIZE provides the same functionality as the Packet Size option; however, SQL_ATTR_PACKET_SIZE and the Packet Size option are mutually exclusive. If Packet Size is specified, the driver returns the message `Driver Not Capable` if an application attempts to call SQLSetConnectAttr() for SQL_ATTR_PACKET_SIZE. If you do not set the Packet Size option, application calls to SQLSetConnectAttr() for SQL_ATTR_PACKET_SIZE are accepted by the driver.
- This connection option can affect performance.

Default

`0`

GUI Tab

Performance tab

See Also

Performance considerations on page 56

Password

Attribute

Password (PWD)

Purpose

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

Valid Values

pwd

where:

pwd

is a valid password.

Default

No default value

GUI Tab

n/a

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

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

Valid Values

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

where:

x

is a positive integer that specifies a number of seconds.

Behavior

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

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

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

Default

`0`

GUI Tab

Advanced tab

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

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

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

Valid Values

`0` | `1` | `2`

Behavior

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

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

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

Default

`0` (Ignore Errors)

GUI Tab

Advanced tab

Select Method

Attribute

SelectMethod (SM)

Purpose

Determines whether database cursors are used for Select statements.

Valid Values

`0` | `1`

Behavior

If set to `0` (Cursor), database cursors are used. In some cases performance degradation can occur when performing large numbers of sequential Select statements because of the amount of overhead associated with creating database cursors.

If set to `1` (Direct), Select statements are run directly without using database cursors, and the data source is limited to one active statement.

Notes

- This connection option can affect performance.

Default

`1` (Direct)

GUI Tab

Performance tab

See Also

Performance considerations on page 56

Server Name

Attribute

InterfacesFileServerName (IFSN)

Purpose

The name of the section in the Interfaces file containing the network connection information for the Sybase IQ server. Typically, the section name is the host name of the Sybase IQ server.

Valid Values

`section_name`

where:

section_name

is a section in the Interfaces file containing the network connection information for the Sybase IQ server.

Notes

- The Network Address option and the Server Name option are mutually exclusive.

Default

No default value

GUI Tab

General tab

Truncate Time Type Fractions

Attribute

TruncateTimeTypeFractions (TTTF)

Purpose

Determines whether the driver sets fractional seconds to zero (0) when converting data from the TIME data type to TIMESTAMP, CHAR, or WCHAR data types.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver converts fractional seconds to zero when converting the TIME data type.

If set to `0` (Disabled), the driver does not set fractional seconds to zero when converting the TIME data type.

Default

`0` (Disabled)

GUI Tab

Advanced tab

User Name

Attribute

LogonID (UID)

Description

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

No default value

GUI Tab

General tab

Workstation ID

Attribute

WorkstationID (WKID)

Purpose

An identifier for the client machine.

Valid Values

ID

where:

ID

is workstation ID use by the client machine.

Default

No default value

GUI Tab

Connection tab