Progress DataDirect

Progress DataDirect for ODBC for Informix Wire Protocol Driver

User's Guide

Release 8.0.2

Copyright

Table of Contents

1

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

The Progress® DataDirect® for ODBC™ for Informix™ Wire Protocol driver (the Informix Wire Protocol driver) supports multiple connections to the Informix Dynamic Server.

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

For details, see the following topics:

- Contacting Technical Support

What's new in this release?

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.
  - For UNIX/Linux, 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. Passwords can be encrypted for any option. See Password Encryption Tool (UNIX/Linux only) on page 43 for details.
- **Changed Behavior**
  - Support has been deprecated for the following versions of Informix:
    - Informix 11.0 and higher
    - Informix 10.0 and higher
    - Informix 9.2 and higher
  - The following Windows platforms have reached the end of their product lifecycle and are no longer supported by the driver:
    - Windows 8.0 (versions 8.1 and higher are still supported)
    - Windows Vista (all versions)
    - Windows XP (all versions)
    - Windows Server 2003 (all versions)

Driver requirements

Data source and platform requirements

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

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

AIX requirements for 32-bit and 64-bit drivers

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

HP-UX requirements for 32-bit drivers

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

HP-UX requirements for 64-bit drivers

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

Oracle Solaris requirements for 32-bit drivers

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

Oracle Solaris requirements for 64-bit drivers

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

Installing and setting up the driver (Windows)

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

To begin accessing data with the driver:

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

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

   c) Follow the prompts to complete installation.

Note:

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

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

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

3. Select either the **User DSN**, **System DSN**, or **File DSN** tab to display a list of data sources.

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

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

4. On the **General** tab of the driver Setup dialog box, provide values for the following essential connection options for user ID/password authentication; then, click **Apply**:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Host Name**: Type the IP address or the host name of the computer that is running the database server to which you want to connect.
   - **Port Number**: Type the port number of the server listener.
   - **Server Name**: Type the name of the database server to which you want to connect.
   - **Database Name**: 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 14 provides a connection string example that can be used to configure common functionality and features. The options and values described in this section apply to all configuration methods.
   - Connection option descriptions on page 53 provides a complete list of supported options by functionality.
   - Performance considerations on page 47 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 Informix 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 37

Installing and setting up the driver (UNIX/Linux)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_nn_LINUX_xx_INSTALL.bin`

   c) Follow the prompts to complete installation.

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

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

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

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

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

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

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

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

[Informix]
Driver=ODBCHOME/lib/xxifcl28.yy
...
HostName=MyInformixHost
...
PortNumber=4321
...
ServerName=Acct
...
Database=Payroll
...
LogonID=jsmith
...
Password=secret
...
```

See Configuration through the system information (odbc.ini) file on page 38 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 37, DSN-less connections, for more information. For examples, see Connection string examples on page 14.

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 14 provides a connection string example that can be used to configure common functionality and features. You can modify this example to create a string that best suites your environment.

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

- Connection option descriptions on page 53 provides a complete list of supported options by functionality.
- Performance considerations on page 47 describes connection options that affect performance, along with recommended settings.

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

This completes the deployment of the driver.

Connection string examples

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

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

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

See also

Using a connection string on page 37

User ID/password authentication

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

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;HostName=host_name;PortNumber=port_number;
ServerName=server_name;Database=database_name;LogonID=user_name;Password=password;
[attribute=value[;...]];
```

where:

host_name

specifies the IP address or the host name of the computer that is running the database server to which you want to connect.

port_number

specifies the port number of the server listener.

server_name

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

database_name

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

user_name

specifies your username.

password

specifies your password.

attribute=value

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

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

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;HostName=MyInformixHost;PortNumber=4321;
ServerName=Acct;Database=Payroll;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 37
Connection option descriptions on page 53

Connection failover

This string configures the driver to use connection failover in conjunction with some of its optional features. It uses the user ID/password authentication method for authentication.

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;AlternateServers=alternate_server;
ConnectionRetryCount=connection_retry_count;ConnectionRetryDelay=connection_retry_delay;
LoadBalancing=load_balancing;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). Set this to `1` to enable load balancing. See "Load Balancing" for details.

user_name

specifies your username.

password

specifies your password.

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

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;AlternateServers=(HostName=informix:
PortNumber=4321:ServerName=Acct:Database=Accounting,HostName=255.201.11.24:
PortNumber=4320:ServerName=Acct1:Database=Accounting);ConnectionRetryCount=4;
ConnectionRetryDelay=5;LoadBalancing=1;LogonID=jsmith;Password=secret;
```

See also

Using a connection string on page 37
Connection option descriptions on page 53
Connection failover on page 46
Load Balancing on page 62

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 ivifcl28.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 Informix data types are mapped to the standard ODBC data types.

Table 1: Informix Data Types

Informix	ODBC
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
BYTE	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB	SQL_LONGVARCHAR

Informix	ODBC
DATE	SQL_TYPE_DATE
DATETIME YEAR TO FRACTION(f) [1]	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO SECOND	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO DAY	SQL_TYPE_DATE
DATETIME HOUR TO SECOND	SQL_TYPE_TIME
DATETIME HOUR TO FRACTION(f)[1]	SQL_TYPE_TIME
DECIMAL	SQL_DECIMAL
FLOAT	SQL_DOUBLE
INT8	SQL_BIGINT
INTEGER	SQL_INTEGER
INTERVAL YEAR(p) TO YEAR	SQL_INTERVAL_YEAR
INTERVAL YEAR(p) TO MONTH	SQL_INTERVAL_YEAR_TO_MONTH
INTERVAL MONTH(p) TO MONTH	SQL_INTERVAL_MONTH
INTERVAL DAY(p) TO DAY	SQL_INTERVAL_DAY
INTERVAL DAY(p) TO HOUR	SQL_INTERVAL_DAY_TO_HOUR
INTERVAL DAY(p) TO MINUTE	SQL_INTERVAL_DAY_TO_MINUTE
INTERVAL DAY(p) TO SECOND	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL DAY(p) TO FRACTION(f)[1]	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL HOUR(p) TO HOUR	SQL_INTERVAL_HOUR
INTERVAL HOUR(p) TO MINUTE	SQL_INTERVAL_HOUR_TO_MINUTE
INTERVAL HOUR(p) TO SECOND	SQL_INTERVAL_HOUR_TO_SECOND
INTERVAL HOUR(p) TO FRACTION(f)[1]	SQL_INTERVAL_HOUR_TO_SECOND
INTERVAL MINUTE(p) TO MINUTE	SQL_INTERVAL_MINUTE
INTERVAL MINUTE(p) TO SECOND	SQL_INTERVAL_MINUTE_TO_SECOND
INTERVAL MINUTE(p) TO FRACTION(f)[1]	SQL_INTERVAL_MINUTE_TO_SECOND

[1] (f) can have a value of 1, 2, 3, 4, or 5. The precision is type-dependent and the scale is 5.

Informix	ODBC
INTERVAL SECOND(p) TO SECOND	SQL_INTERVAL_SECOND
INTERVAL SECOND(p) TO FRACTION(f)[1]	SQL_INTERVAL_SECOND
LVARCHAR(p) [2]	SQL_VARCHAR
MONEY	SQL_DECIMAL
NCHAR	SQL_CHAR
NVARCHAR	SQL_VARCHAR
SERIAL	SQL_INTEGER
SERIAL8	SQL_BIGINT
SMALLFLOAT	SQL_REAL
SMALLINT	SQL_SMALLINT
TEXT	SQL_LONGVARCHAR
VARCHAR	SQL_VARCHAR

See Retrieving data type information on page 20 for more information about data types.

Retrieving data type information

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

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

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

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

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

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

    SQLINTEGER      strlenTypeName,
```

[2] Supported only on Informix 9.4 and higher servers.

```
                    strlenSqlDataType,
                    strlenColumnSize;

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

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

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

        do {

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

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

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

Driver specifications

This section describes the general functionality supported by the driver.

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

  In addition, the following functions are supported:

  - SQLColumnPrivileges
  - SQLForeignKeys
  - SQLTablePrivileges

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

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

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

- **Connections and statements supported**: The driver supports multiple connections and multiple statements per connection.
- **SQL Support**: The driver supports the minimum SQL grammar.

Additional information

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

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

Troubleshooting

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

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

Contacting Technical Support

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

https://www.progress.com/support

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

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

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

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

November 2021, Release 8.0.2 for the Progress DataDirect for ODBC for Informix 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 Informix.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.

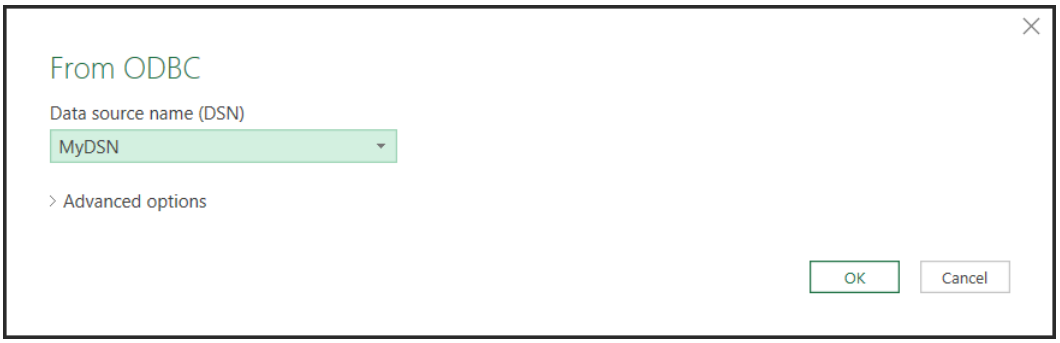
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

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.

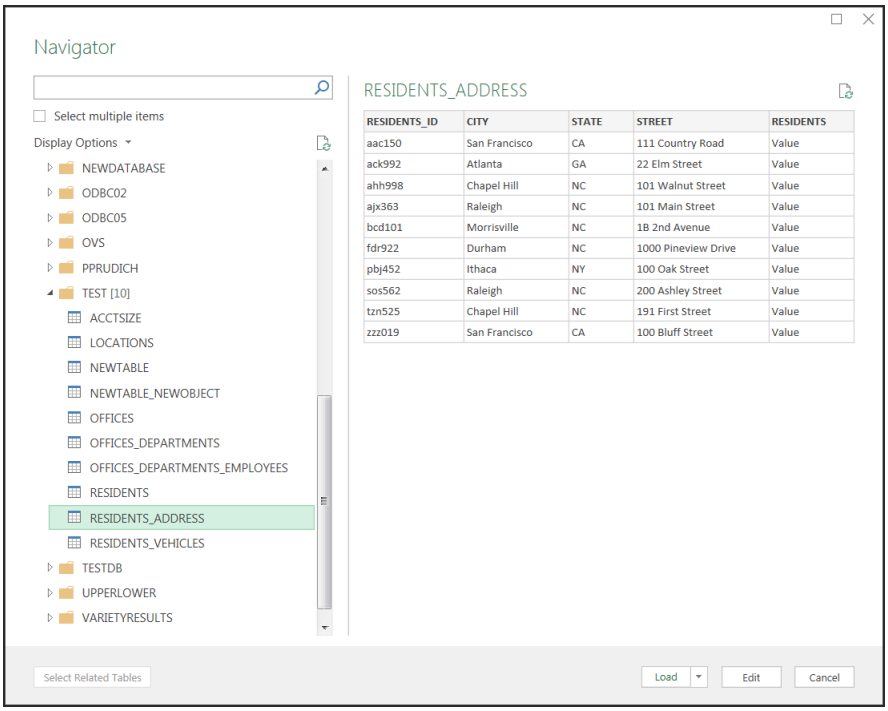
To use the driver to access data with Excel from the Data Connection Wizard:

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.

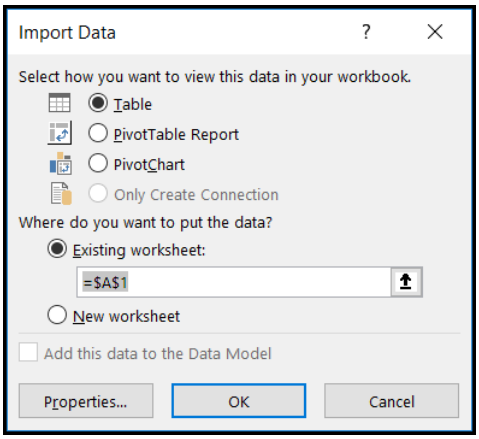


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

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
- Connection failover

- Performance considerations

Environment settings

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

UNIX/Linux environment variables

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

Library search path

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

```
echo $SHELL
```

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

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

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

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

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

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

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

```
env
```

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

ODBCINI

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

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

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

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

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

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

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

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

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

ODBCINST

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

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

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

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

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

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

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

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

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

DD_INSTALLDIR

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

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

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

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

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

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

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

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

The next step is to test load the driver.

The Test Loading Tool

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

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

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

```
ivtestlib/opt/odbc/lib/ivifclxx.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 an Informix data source:

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

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

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

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

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

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

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

3. On the General tab, specify values for the following options:

   - **Data Source Name**: Type a string that identifies this data source configuration, such as `Projects`.
   - **Description**: Type an optional long description of a data source name, such as `My Development Projects`.
   - **Host Name**: Type the IP address or the host name of the computer that is running the database server to which you want to connect.
   - **Port Number**: Type the port number of the server listener.

- **Server Name**: Type the name of the database server to which you want to connect.
- **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 36, provide values for options to configure advanced behavior.
6. If applicable, on the Failover tab on page 37, configure failover data source settings.
7. 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.

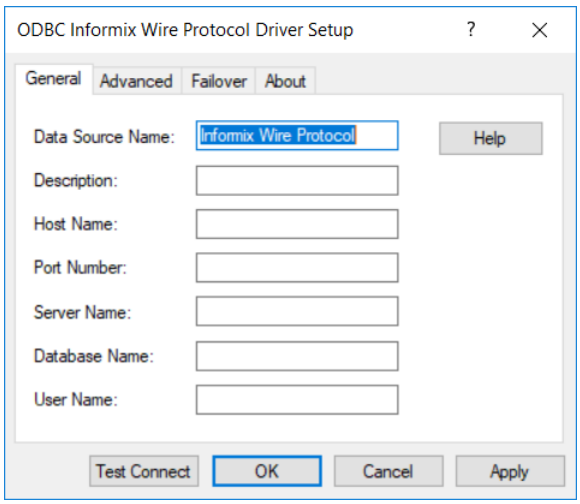
See also

Using a logon dialog box on page 44

General tab

The General tab allows you to configure essential and required options that are used to create a data source.

Figure 1: General tab



Connection Options: General	Default
Data Source Name on page 60	No default value
Description on page 60	No default value
Host Name on page 61	No default value
Port Number on page 63	No default value
Server Name on page 64	No default value
Database Name on page 59	No default value
User Name on page 66	No default value

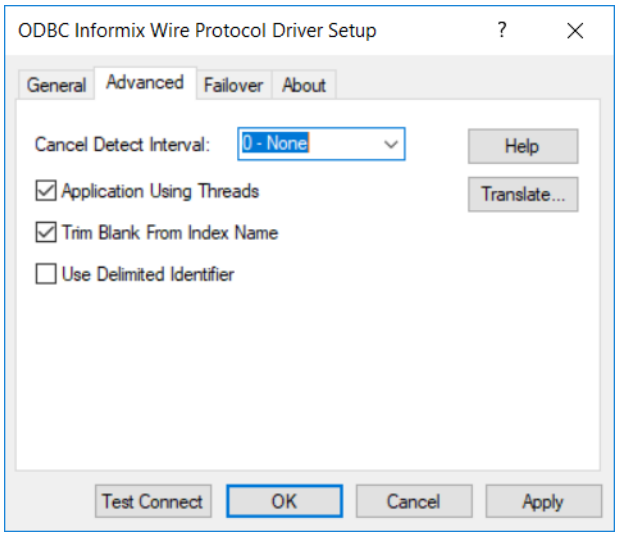
See also

Configuring the driver using the GUI on page 33

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
Cancel Detect Interval on page 57	**0-None**
Application Using Threads on page 56	Enabled
Trim Blank From Index on page 64	Enabled
Use Delimited Identifier on page 65	Disabled

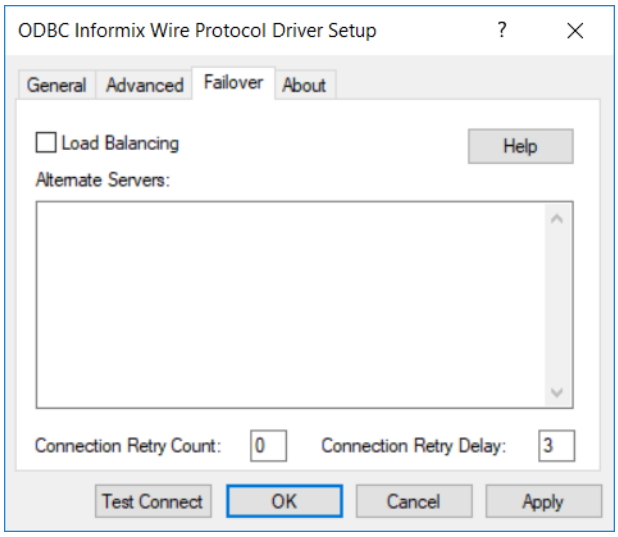
See also

Configuring the driver using the GUI on page 33

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 3: Failover tab



Connection Options: Failover	Default
Load Balancing on page 62	Disabled
Alternate Servers on page 55	No default value
Connection Retry Count on page 58	`0`
Connection Retry Delay on page 58	3

See also

Configuring the driver using the GUI on page 33

Using a connection string

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

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

The DSN-less connection string has the form:

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

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

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

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

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

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

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;HOST=MyInformixHost;
PORT=4321;SRVR=ACCT;DB=Infsdb1;UID=JOHN;PWD=XYZZY
```

See also

Connection option descriptions on page 53
Connection string examples on page 14

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:

```
Informix=DataDirect 8.0 Informix 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, [Informix 2]. The data source definitions contain connection string *attribute=value* pairs with default values. You can modify these values as appropriate for your system. "Connection Option Descriptions" describes these attributes. See "Sample Default odbc.ini File" for sample data sources.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

See also

Connection option descriptions on page 53
Sample default odbc.ini file on page 40
File data sources on page 42
IANAAppCodePage on page 61
DSN-less connections on page 41

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]
Informix=DataDirect 8.0 Informix Wire Protocol

[Informix]
Driver=ODBCHOME/lib/ivifcl28.so
Description=DataDirect 8.0 Informix Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
CancelDetectInterval=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=
HostName=
LoadBalancing=0
LogonID=
Password=
PortNumber=
ServerName=
TrimBlankFromIndexName=1
UseDelimitedIdentifier=0

[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, `[Informix]`.

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 53

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 Informix 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 31
Configuration through the system information (odbc.ini) file on page 38

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 Informix Wire Protocol=Installed

[DataDirect 8.0 Informix Wire Protocol]
Driver=ODBCHOME/lib/ivifcl28.so
APILevel=0
ConnectFunctions=YYY
CPTimeout=60
DriverODBCVer=3.52
FileUsage=0
SQLLevel=0
UsageCount=1

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

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

File data sources

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

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

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

```
[ODBC]
Driver=DataDirect 8.0 Informix Wire Protocol
...
HostName=MyInformixHost
...
PortNumber=4321
...
ServerName=Acct
...
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\Informix2.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/Informix2.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/Informix2.dsn;UID=john;PWD=test01
```

See also

Configuring and connecting to data sources on page 29
DSN-less connections on page 41
Additional configuration methods for UNIX and Linux on page 38

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`*

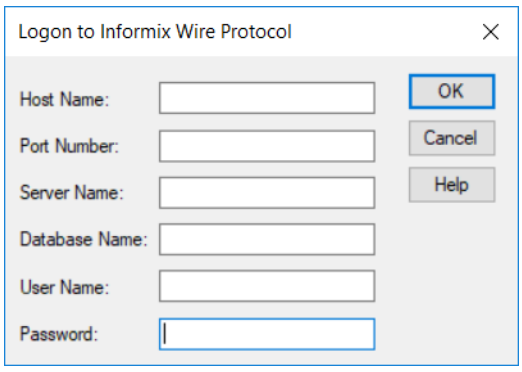
4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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 4: Logon to Informix dialog box



In this dialog box, provide the following information:

1. In the Host Name field, type either the IP address or the host name of the computer that is running the database server to which you want to connect.
2. In the Port Number field, type the port number of the server listener.
3. In the Server Name field, type the name of the database server
4. In the Database Name field, type the name of the database to which you want to connect.
5. In the User Name field, type your Informix user name.
6. In the Password field, type your password.
7. Click **OK** to complete the logon.

Authentication

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

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

- Set the Host Name (HostName) option to specify the IP address or the host name of the computer that is running the database server to which you want to connect.
- Set the Port Number (PortNumber) option to specify the port number of the server listener.
- Set the Server Name (ServerName) option to specify the name of the database server to which you want to connect.
- Set the Database Name (Database) option to specify the name of the database to which you want to connect.
- Set the User Name (LogonID) option to specify your user name.
- Set the Password option to specify your password.

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

Connection string

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;HostName=myserver;PortNumber=4321;
ServerName=Acct;Database=Payroll;LogonID=John;Password=secret
```

odbc.ini

```
[Informix]
Driver=ODBCHOME/lib/xxifcl28.yy
...
HostName=myserver
...
PortNumber=4321
...
ServerName=Acct
...
Database=Payroll
...
LogonID=John
...
Password=secret
...
```

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

Connection failover

The driver 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.

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.

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

To configure failover:

- 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).
- 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.
- 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.
- 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 configure the driver to use connection failover in conjunction with some of its optional features. It uses the user ID/password authentication method for authentication.

Connection string

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;AlternateServers=(HostName=informix:
PortNumber=4321:ServerName=Acct1:Database=Accounting,HostName=255.201.11.24:
PortNumber=4320:ServerName=Acct2:Database=Accounting);ConnectionRetryCount=4;
ConnectionRetryDelay=5;LoadBalancing=1;LogonID=John;Password=secret;
```

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

odbc.ini

```
Driver=ODBCHOME/lib/ivifclxx.so
Description=DataDirect Informix Wire Protocol
...
AlternateServers=(HostName=informix:PortNumber=4321:ServerName=Acct1:
Database=Accounting,HostName=255.201.11.24:PortNumber=4320:
ServerName=Acct2:Database=Accounting)
...
ConnectionRetryCount=4
...
ConnectionRetryDelay=5
...
LoadBalancing=0
...
LogonID=John;
...
Password=secret;
...
```

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), and to attempt reconnecting on new connections only.

Note: The LogonID and Password options are not required to be stored in the data source or 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 data source or connection string.

See also

Guidelines for primary and alternate servers on page 47
Guidelines for primary and alternate servers on page 47
Connection option descriptions on page 53

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.

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.

Cancel Detect Interval (CancelDetectInterval): If your application uses threads, it may allow canceling of long running queries (may issue synchronous SQLCancel calls). If your application does not issue synchronous SQLCancel calls, the driver can improve performance if the CancelDetectInterval attribute is disabled (set to 0). In this case, the driver does not incur the overhead of periodically checking for SQLCancel. In the case where your application does issue synchronous SQLCancel calls, this attribute should be set to a value that specifies how often the driver checks to see if a long running query has been canceled.

4

Additional features and functionality

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

For details, see the following topics:

- Using IP addresses
- Persisting a result set as an XML data file
- MTS Support

Using IP addresses

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

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

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;
HOST=123.456.78.90;PORT=4321;SRVR=ACCT;
DB=PSQLACCT;UID=JOHN;PWD=XYZZYYou
```

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;
HOST=2001:DB8:0000:0000:8:800:200C:417A;PORT=4321;
SRVR=ACCT;DB=SQLSACCT;UID=JOHN;PWD=XYZZYYou
```

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

```
DRIVER=DataDirect 8.0 Informix Wire Protocol;
HOST=2001:DB8:0:0:8:800:200C:417A;PORT=4321;
SRVR=ACCT;DB=SQLSACCT;UID=JOHN;PWD=XYZZYYou
```

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

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

Persisting a result set as an XML data file

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

1. Turn on STATIC cursors. For example:

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

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

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

2. Execute a SQL statement. For example:

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

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

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

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

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

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

```
Function Sequence Error
```

Using the Windows XML Persistence Demo tool

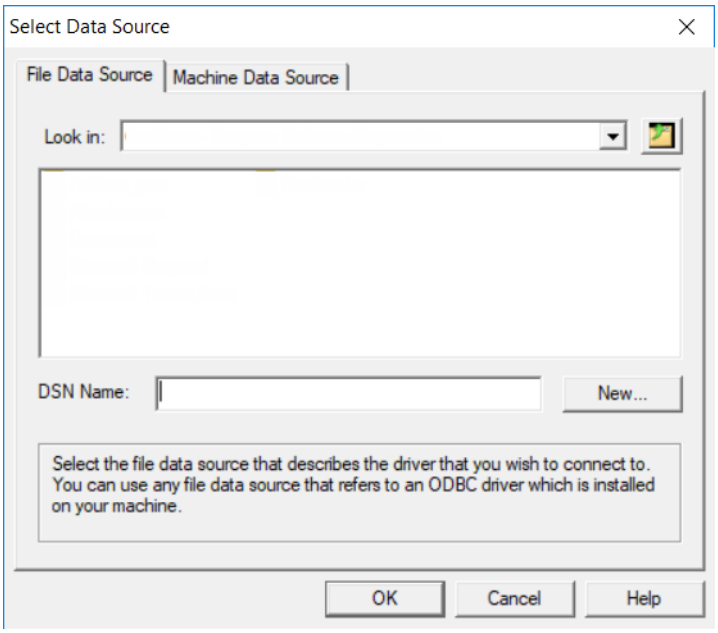
The driver for Windows is shipped with an XML persistence demo tool. This tool is installed in the product installation directory.

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

To use the Windows XML Persistence Demo tool:

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

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



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

- Create a new machine data source by clicking the **Machine Data Source** tab and clicking **New**. The Create New Data Source dialog box appears. Follow the instructions in the dialog box.

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

   Note that the Status box in the XML Persistence dialog box displays whether the action failed or succeeded.
6. Click **Disconnect** to disconnect from the database.
7. Click **Close** to exit the tool.

Using the UNIX/Linux XML Persistence Demo tool

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

MTS Support


On Windows, the driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC) using the XA Protocol. For a general discussion of MTS and DTC, refer to the help file of the Microsoft Transaction Server SDK.

Note: The 32-bit driver can operate in a 64-bit Windows environment; however, it does not support DTC in this environment. Only the 64-bit driver supports DTC in a 64-bit Windows environment.

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.

Required options

The following table summarizes options that are required for authenticating to the server.

Table 2: Required options

Attribute (Short Name)	Default
Database (DB)	No default value
DataSourceName (DSN)	No default value
Description (n/a)	No default value
HostName (HOST)	No default value
LoginID (UID)	No default value
Password (PWD)	No default value
PortNumber (PORT)	No default value
ServerName (SRVR)	No default value

Failover options

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

Table 3: Failover options

Attribute (Short Name)	Default
AlternateServers (ASRV)	No default value
ConnectionRetryCount (CRC)	`0`
ConnectionRetryDelay (CRD)	`3`
LoadBalancing (LB)	`0` (Disabled)

Additional options

The following table summarizes additional options.

Table 4: Additional options

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	`1` (Enabled)
CancelDetectInterval (CDI)	`0` (None)
IANAAppCodePage (IACP) UNIX/Linux only	`4` (ISO 8559-1 Latin-1)

Attribute (Short Name)	Default
TrimBlankFromIndexName (TBFIN)	1 (Enabled)
UseDelimitedIdentifier (UDI)	0 (Disabled)

For details, see the following topics:

- Alternate Servers
- Application Using Threads
- Cancel Detect Interval
- Connection Retry Count
- Connection Retry Delay
- Database Name
- Data Source Name
- Description
- Host Name
- IANAAppCodePage
- Load Balancing
- Password
- Port Number
- Server Name
- Trim Blank From Index
- Use Delimited Identifier
- User Name

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

```
(Database=databasename:HostName=hostvalue:
PortNumber=portvalue:ServerName=servervalue[, . . .])
```

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

Example

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

```
(Database=Infdb1:HostName=Informixhost1:PortNumber=5177:ServerName=accounting1,
Database=Infdb2:HostName=Informixhost2:PortNumber=5178:ServerName=accounting2)
```

Notes

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

Default

None

GUI Tab

Failover tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

This connection option can affect performance.

Valid Values

`0` | `1`

Behavior

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

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

Default

`1` (Enabled)

GUI Tab

Advanced tab

See also

- Performance considerations on page 47

Cancel Detect Interval

Attribute

CancelDetectInterval (CDI)

Purpose

Determines whether long-running queries in threaded applications can be cancelled if the application issues a SQLCancel.

This connection option can affect performance.

Valid Values

`0` | *x*

where:

x

is the number of seconds the driver waits before checking for SQLCancel calls.

Behavior

If set to `0` (None), the driver does not allow long-running queries in threaded applications to be canceled, even if the application issues a SQLCancel.

If set to x (seconds), for every pending query, the driver checks for SQLCancel calls at the specified interval. If the driver determines that a SQLCancel has been issued, the driver cancels the query.

Example

If you specify `5`, for every pending query, the driver checks every five seconds to see whether the application has issued a SQLCancel call. If it detects a SQLCancel call, the driver cancels the query.

Default

`0` (None)

GUI Tab

Advanced tab

See also

- Performance considerations on page 47

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

See 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

See Failover tab

Database Name

Attribute

Database (DB)

Purpose

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

Valid Values

database_name

where:

database_name

is the name of a valid database.

Default

None

GUI Tab

General 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

None

GUI Tab

General tab

Description

Attribute

Description (n/a)

Purpose

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

Valid Values

`string`

where:

`string`

is a description of a data source.

Default

None

GUI Tab

General tab

Host Name

Attribute

HostName (HOST)

Purpose

The host name or the IP address of the computer that is running the database server to which you want to connect.

Valid Values

host_name | *IP_address*

where:

host_name

is the host name of the computer running the database server.

IP_address

is the IP address of the computer running the database server.

The IP address can be specified in either IPv4 or IPv6 format. For details about these formats, see Using IP addresses on page 49.

Default

None

GUI Tab

General 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 Manager checks 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`)

Valid Values

`IANA_code_page`

where:

`IANA_code_page`

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

Notes

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

Default

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

GUI Tab

Advanced tab

Load Balancing

Attribute

LoadBalancing (LB)

Description

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

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

password

where:

password

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

The port number of the server listener.

Valid Values

port_number

where:

port_number

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

Default

None

GUI Tab

General tab

Server Name

Attribute

ServerName (SRVR)

Purpose

Specifies the name of the Informix server.

Valid Values

server_name

where:

server_name

is a name that uniquely identifies the Informix server.

Default

None

GUI Tab

General tab

Trim Blank From Index

Attribute

TrimBlankFromIndexName (TBFIN)

Purpose

Determines whether the driver trims leading spaces from system-generated index names. Some applications cannot process a leading space in index names.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver trims leading spaces from system-generated index names.

If set to 0 (Disabled), the driver does not trim leading spaces from system-generated index names.

Default

1 (Enabled)

GUI Tab

Advanced tab

Use Delimited Identifier

Attribute

UseDelimitedIdentifier (UDI)

Purpose

Determines whether the driver sets the Informix DELIMIDENT environment variable. The DELIMIDENT environment variable specifies that strings enclosed between double quotation marks (") are delimited database identifiers.

Valid Values

0 | 1

Behavior

If set to 1 (enabled), the Informix server interprets strings enclosed in double quotation marks as identifiers, not as string literals.

If set to 0 (disabled), the Informix server interprets strings enclosed in double quotation marks as string literals, not as identifiers.

Default

0 (Disabled)

GUI Tab

Advanced tab

User Name

Attribute

LogonID (UID)

Purpose

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

Valid Values

userid

where:

userid

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

Default

None

GUI Tab

General tab