Progress DataDirect Connect Series for ODBC

User's Guide

Release 7.1.6

Copyright

Visit the following page online to see Progress Software Corporation’s current Product Documentation Copyright Notice/Trademark Legend: https://www.progress.com/legal/documentation-copyright.

Updated: 2026/05/08

Table of Contents

1

Welcome to the Progress DataDirect Connect Series for ODBC

The Progress® DataDirect Connect® Series *for* ODBC™ provides ODBC drivers for a number of leading databases, as well as flat-file database systems. The drivers are compliant with the Open Database Connectivity (ODBC) specification and compatible with ODBC 3.8 applications. The drivers are tested and supported across numerous platforms. Progress DataDirect Connect Series *for* ODBC includes the following products:

- DataDirect Connect *for* ODBC
- DataDirect Connect64 *for* ODBC
- DataDirect Connect XE (Extended Edition) *for* ODBC
- DataDirect Connect64 XE *for* ODBC

Note: This guide contains information for the 7.1.6 version of the ODBC drivers. For 8.0 and higher version of the drivers, documentation is available at the Progress DataDirect Connectors Documentation page: https://docs.progress.com/bundle/datadirect-connectors/page/DataDirect-Connectors-by-data-source.html.

The content of this guide assumes that you are familiar with your operating system and its commands. It contains the following information:

- Quick Start Connect on page 23 explains the basics for quickly configuring and testing the drivers.
- General Information on Using Connect Drivers on page 37 explains the drivers and ODBC, and discusses environment-specific subjects.
- Advanced Features on page 55 explains at a general level advanced driver features such as failover, security, connection pooling, and bulk load.

- Configuring the Product on UNIX/Linux on page 87 discusses UNIX and Linux environment variables and configuration of the drivers. It also provides a sample system information file, as well as discussing other driver tools for UNIX and Linux.
- A chapter for each database driver. Each driver's chapter is structured in the same way. First, it lists which versions of the databases the driver supports, the operating environments in which the driver runs, and the driver requirements for your operating environment. Next, it explains how to configure a data source and how to connect to that data source. Finally, the chapter provides information about data types, ODBC conformance levels, isolation and lock levels supported, and other driver-specific information.

The documentation for DataDirect Connect Series *for* ODBC drivers 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.

If you are writing programs to access ODBC drivers, you need to obtain a copy of the *ODBC Programmer's Reference* for the Microsoft Open Database Connectivity Software Development Kit, available from Microsoft Corporation.

For the latest information about the specific drivers available for your platform, refer to the readme file in your software package.

Database drivers are continually being added to each operating environment. For the latest information about the specific drivers available for your platform, refer to the readme file in your software package, or refer to the DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

Note: This guide refers the reader to Web pages using URLs for more information about specific topics, including Web URLs not maintained by Progress DataDirect. Because it is the nature of Web content to change frequently, Progress DataDirect can guarantee only that the URLs referenced in this guide were correct at the time of publication.

For details, see the following topics:

- What's new in this release?
- Product Matrix
- Product Platforms
- About the Documentation Library
- Contacting Technical Support

What's new in this release?

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

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

End of Life Announcement: The following drivers have reached the end of their product life cycle and are no longer supported:

- Informix Client
- Oracle Client
- SQL Server Legacy

End of Support for the 7.1 Version: The 7.1 version of the following drivers is no longer supported:

- Driver for Apache Hive
- Db2 Wire Protocol
- Informix Wire Protocol
- MySQL Wire Protocol Driver
- Oracle Wire Protocol Driver
- PostgreSQL Wire Protocol
- Salesforce
- SQL Server Wire Protocol
- Sybase IQ Wire Protocol

Note that connectivity to Apache Hive, Db2, Informix, MySQL, Oracle, PostgreSQL, Salesforce, SQL Server, and Sybase IQ data sources is still supported by version 8.0 of the wire protocol drivers.

The highlights of this release are:

- **Support for setting the value of undocumented connection options using the setup dialog for the following drivers on Windows:**
  - Greenplum Wire Protocol
  - Impala Wire Protocol
  - Progress OpenEdge Wire Protocol
  - Sybase Wire Protocol
- A Password Encryption Tool, called ddencpwd, is now included with the product package. It encrypts passwords for secure handling in connection strings and odbc.ini files. At connection, the driver decrypts these passwords and passes them to the data source as required. See Password Encryption Tool (UNIX/Linux only) on page 100 for details.
- **Greenplum driver enhancements**
  - Support has ended for the DataDirect ODBC Data Source Administrator for Linux. As a result, the Linux ODBC Administrator will no longer be installed with the product. In addition, to avoid exposure to potential security vulnerabilities, the installer program will remove Linux ODBC Administrator files from existing installation directories during updates and new product installations.
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

    The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL Server Authentication on page 67 and TLS/SSL Client Authentication on page 68 for details.

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

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

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

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

- The crypto protocol versions prior to TLSv1 are no longer supported.
- The driver has been enhanced with the new Batch Mechanism (BatchMechanism) connection option, which specifies the preferred mechanism for executing batch insert operations. By setting Batch Mechanism to 2 (MultiRowInsert) or 3 (Copy), the driver can achieve substantial performance gains when performing batch inserts. The default setting is `BatchMechanism=1`. See Batch Mechanism on page 342 for details.
- The driver has been enhanced to support the following data types: Citext, Float, and Tinyint. See Data Types on page 380 for details.
- A Power BI connector is now included with the product package. You can use this connector to access your Greenplum data with Power BI. See Accessing Greenplum data with Power BI on page 336 for details.
- The new AllowedOpenSSLVersions option allows you to determine which version of the OpenSSL library file the driver uses for data encryption.
- Added support for Kerberos authentication, including the following connection options:
  - The Authentication Method connection option now allows you to specify the method the driver uses to authenticate the user to the server when a connection is established. See Authentication Method on page 341 for details.
  - The GSS Client Library connection option now allows you to specify the name of the GSS client library that the driver uses to communicate with the Key Distribution center (KDC). See GSS Client Library on page 356 for details.
  - The Service Principal Name connection option allows you to specify the service principal name to be used by the driver for Kerberos authentication. See Service Principal Name on page 371 for details.
- The new CryptoLibName and SSLLibName connection options allow you to designate the OpenSSL libraries used when SSL is enabled. See CryptoLibName on page 346 and SSLLibName on page 372 for details.
- The new Crypto Protocol Version connection option allows you to specify the cryptographic protocols used when SSL is enabled. See Crypto Protocol Version on page 345 for details.

- The new Unbounded Numeric Precision connection option allows you to define the precision for unbounded NUMERIC columns when described within the column, parameter, result set, or table metadata. See Unbounded Numeric Precision on page 377 for details.
- The new Unbounded Numeric Scale connection option allows you to define the scale for unbounded NUMERIC columns described within the column, parameter, result set, or table metadata. See Unbounded Numeric Scale on page 377 for details.
- The TCP Keep Alive connection option allows you to use TCP Keep Alive to maintain idle TCP connections. See TCP Keep Alive on page 373 for details.
- The Max Char Size connection option specifies the maximum size of columns of type SQL_VARCHAR that the driver describes through result set descriptions and catalog functions. See Max Char Size on page 365 for details.
- The Max Long Varchar Size connection option specifies the maximum size of columns of type SQL_LONGVARCHAR that the driver describes through result set descriptions and catalog functions. See Max Long Varchar Size on page 365 for details.
- The Enable Keyset Cursors connection option enables emulated Keyset cursors to provide scrollable cursors to an ODBC application. See Enable Keyset Cursors on page 349 for details.
- The Keyset Cursor Options connection option determines which columns are used to comprise the keyset that the driver uses to create the initial keyset on which cursor operations are based. See Keyset Cursor Options on page 360 for details.
- Added support for SSL encryption with Greenplum 4.2 and higher, including the following connection options:
  - The Encryption Method connection option now allows you to encrypt data sent between the driver and the database. See Encryption Method on page 351 for details.
  - The Host Name In Certificate connection option now allows you to specify the host name for certificate validation when SSL encryption and validation are enabled. See Host Name In Certificate on page 358 for details.
  - The Key Password connection option now allows you to specify the key password that is used to access the individual keys in the keystore file when SSL and SSL client authentication are enabled on the database server. See Key Password on page 361 for details.
  - The Keystore connection option now allows you to specify the directory containing the keystore file that is to be used when SSL and SSL client authentication are enabled on the database server. See Keystore on page 361 for details.
  - The Key Store Password connection option allows you to specify the keystore password that is used to access the keystore file when SSL and SSL client authentication are enabled on the database server. See Keystore Password on page 362 for details.
  - The Truststore connection option now allows you to specify the directory that contains the truststore file and the truststore file name that is to be used when SSL is enabled and the server authentication is used. See Truststore on page 374 for details.
  - The Truststore Password connection option now allows you to specify the truststore password that is used to access the truststore file when SSL is enabled and the server authentication is used. See Truststore Password on page 375 for details.
  - The User Name connection option now allows you to specify the user ID that is used to connect to your database. See User Name on page 375 for details.
  - The Validate Server Certificate connection option now determines whether the driver validates the certificates that are sent by the database server when SSL encryption is enabled. See Validate Server Certificate on page 376 for details.

- **Impala driver enhancements since General Availability**
  - The default version of the OpenSSL library has been upgraded to 3.5.6. As part of this upgrade, earlier version of the OpenSSL 3.0 library continues to be supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files:
    - Windows: `ivopenssl.dll` and `ddopenssl.dll`
    - Unix: `ivopenssl.so` and `ddopenssl.so`
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

    The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL Server Authentication on page 67 and TLS/SSL Client Authentication on page 68 for details.
  - The Allowed OpenSSL Versions (AllowedOpenSSLVersions) connection option has been deprecated.
  - The product no longer includes version 1.1.1 of the OpenSSL library. The library will reach the end of its product life cycle in September 2023 and will not receive any security updates after that. Note that continuing to use the library after September 2023 can potentially expose you to security vulnerabilities.

    Note: As a result of this change, when installing a new version of the product, the installer program will automatically remove version 1.1.1 of the library from the install directory, which will impact all the DataDirect ODBC drivers installed on a machine. Therefore, if you are using multiple drivers, upgrade all your drivers to the latest version.
  - The product no longer includes version 1.0.2 of the OpenSSL library. The library has reached the end of its product life cycle and is not receiving security updates anymore. Note that continuing to use the library could potentially expose you to security vulnerabilities.

    Note: As a result of this change, when installing a new version of the driver, the installer program will automatically remove version 1.0.2 of the library from the install directory.
  - The crypto protocol versions prior to TLSv1 are no longer supported.
  - The BatchMechanism connection option has been added to the driver. When BatchMechanism is set to 2 (MultiRowInsert), the driver executes a single insert for all the rows contained in a parameter array. MultiRowInsert is the default setting and provides substantial performance gains when performing batch inserts. See Batch Mechanism on page 395 for details.
  - Added support for SSL encryption, including the following connection options:
    - The CryptoLibName connection option allows you to determine the cryptographic library used when SSL is enabled. See CryptoLibName on page 396 for details.
    - Crypto Protocol Version connection allows you to specify the cryptographic protocols used when SSL is enabled. See Crypto Protocol Version on page 396 for details.
    - The Encryption Method connection option now allows you to encrypt data sent between the driver and the database. See Encryption Method on page 402 for details.
    - The Host Name In Certificate connection option now allows you to specify the host name for certificate validation when SSL encryption and validation are enabled. See Host Name In Certificate on page 404 for details.

- The Key Password connection option now allows you to specify the key password that is used to access the individual keys in the keystore file when SSL and SSL client authentication are enabled on the database server. See Key Password on page 405 for details.
- The Key Store connection option now allows you to specify the directory containing the keystore file that is to be used when SSL and SSL client authentication are enabled on the database server. See Key Store on page 405 for details.
- The Keystore Password connection option allows you to specify the keystore password that is used to access the keystore file when SSL and SSL client authentication are enabled on the database server. See Keystore Password on page 406 for details.
- The SSLLibName connection option allows you to determine the SSL library used when SSL is enabled. See SSLLibName on page 412 for details.
- The Truststore connection option now allows you to specify the directory that contains the truststore file and the truststore file name that is to be used when SSL is enabled and the server authentication is used. See Truststore on page 416 for details.
- The Trust Store Password connection option now allows you to specify the truststore password that is used to access the truststore file when SSL is enabled and the server authentication is used. See Trust Store Password on page 415 for details.
- The User Name connection option now allows you to specify the user ID that is used to connect to your database. See User Name on page 417 for details.
- The Validate Server Certificate connection option now determines whether the driver validates the certificates that are sent by the database server when SSL encryption is enabled. See Validate Server Certificate on page 418 for details.

- The Authentication Method connection option has been refreshed with a new valid value for enabling Kerberos Authentication. To use Kerberos authentication with the driver, set `AuthenticationMethod=4`. See Authentication Method on page 394 for details.

  Note: The legacy setting for enabling Kerberos Authentication (`AuthenticationMethod=1`) will continue to be valid for this release; however, it will be disabled in future versions of the product.

- Support for Kerberos Authentication, which can be configured using the following connection options:
  - Authentication Method specifies the method the driver uses to authenticate the user to the server when a connection is established. See Authentication Method on page 394 for details.
  - GSS Client Library specifies the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC). See GSS Client Library on page 402 for details.
  - Service Principal Name specifies the service principal name to be used by driver for Kerberos authentication. See Service Principal Name on page 411 for details.
- Certified with Apache Sentry for Impala 1.1 and higher. Sentry enables administrators to control access to data and metadata stored on an Hadoop cluster by defining user roles and permissions. See Apache Sentry on page 420 for details.
- The driver has been enhanced to support the Char, Decimal, and Varchar data types when connected to Impala 2.0 and higher. See Data Types on page 419 for details.
- The Array Size configuration option has been refreshed to allow specifying the number of cells retrieved instead of rows. By determining the fetch size based on the number of cells, the driver can avoid out of memory errors when fetching from tables containing a large number of columns. See Array Size on page 394 for details.

- **Progress OpenEdge driver enhancements**
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

    The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL Server Authentication on page 67 and TLS/SSL Client Authentication on page 68 for details.
  - The Allowed OpenSSL Versions (AllowedOpenSSLVersions) connection option has been deprecated.
  - The product no longer includes version 1.1.1 of the OpenSSL library. The library will reach the end of its product life cycle in September 2023 and will not receive any security updates after that. Note that continuing to use the library after September 2023 can potentially expose you to security vulnerabilities.

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

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

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

  - The crypto protocol versions prior to TLSv1 are no longer supported.
  - The new AllowedOpenSSLVersions option allows you to determine which version of the OpenSSL library file the driver uses for data encryption.
  - The new CryptoLibName and SSLLibName connection options allow you to designate the OpenSSL libraries used when SSL is enabled. See CryptoLibName on page 115 and Encryption Method on page 120 for details.
  - The new Crypto Protocol Version connection option allows you to specify the cryptographic protocols used when SSL is enabled. See Crypto Protocol Version on page 114 for details.
  - The TCP Keep Alive connection option allows you to use TCP Keep Alive to maintain idle TCP connections. See TCP Keep Alive on page 131 for details.
- **Sybase Wire Protocol driver enhancements**
  - Support has ended for the DataDirect ODBC Data Source Administrator for Linux. As a result, the Linux ODBC Administrator will no longer be installed with the product. In addition, to avoid exposure to potential security vulnerabilities, the installer program will remove Linux ODBC Administrator files from existing installation directories during updates and new product installations.
  - The Performance Wizard is no longer supported.
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL Server Authentication on page 67 and TLS/SSL Client Authentication on page 68 for details.

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

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

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

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

- The crypto protocol versions prior to TLSv1 are no longer supported.
- The driver has been enhanced to support Sybase Extended Password Encryption and Sybase Extended Plus Encrypted Password, which use the asymmetrical key type. This provides stronger password encryption for the secure transmission of public key passwords over networks. See Authentication Method on page 159 for details.
- The driver has been enhanced to support BINARY and VARBINARY data types when using the bulk load protocol.
- The new AllowedOpenSSLVersions option allows you to determine which version of the OpenSSL library file the driver uses for data encryption.
- The new CryptoLibName and SSLLibName connection options allow you to designate the OpenSSL libraries used when SSL is enabled. See CryptoLibName on page 166 and Crypto Protocol Version on page 165 for details.
- The new Crypto Protocol Version connection option allows you to specify the cryptographic protocols used when SSL is enabled. See Crypto Protocol Version on page 165 for details.
- The TCP Keep Alive connection option allows you to use TCP Keep Alive to maintain idle TCP connections. See TCP Keep Alive on page 199 for details.

- **Driver for the Teradata Database enhancements**
  - Support has ended for the DataDirect ODBC Data Source Administrator for Linux. As a result, the Linux ODBC Administrator will no longer be installed with the product. In addition, to avoid exposure to potential security vulnerabilities, the installer program will remove Linux ODBC Administrator files from existing installation directories during updates and new product installations.
  - The driver has been enhanced to support the Number data type. See Data Types on page 449 for details.

- **Text driver enhancements**

- Support has ended for the DataDirect ODBC Data Source Administrator for Linux. As a result, the Linux ODBC Administrator will no longer be installed with the product. In addition, to avoid exposure to potential security vulnerabilities, the installer program will remove Linux ODBC Administrator files from existing installation directories during updates and new product installations.
- The driver has been enhanced to support a maximum character length of 255 characters for table and column names.

- **XML driver enhancements**
  - The driver has been enhanced to support Microsoft XML parser (MSXML) 6.0. Note that MSXML 6.0 (msxml6.dll) must be installed on your machine for the driver to function correctly.

- **New driver DataDirect Connect64 *for* ODBC**
  - Text Driver: A 64-bit version of the driver is now generally available. This driver provides 64-bit support for the features and functionality offered by the earlier DataDirect Connect (32-bit) version of the driver. In addition, for the Connect64 driver, support for HP-UX IPF has been added. See The Text Driver on page 213 for details.

    Note: The Connect64 Text driver uses a driver-specific installer, instead of the Connect Series installer. Both installers are available on the Progress website.

- **New driver DataDirect Connect XE and DataDirect Connect64 XE *for* ODBC**
  - Impala™ Wire Protocol Driver
    - Supports Cloudera Impala database servers and formally certified with the following file formats and storage handlers:
      - File Formats:
        - Parquet
        - TextFile
      - Storage Handlers:
        - HBase
    - Returns result set metadata for parameterized statements that have been prepared by not yet executed.
    - Supports parameter arrays, processing the arrays as a series of executions, one execution for each row in the array.
    - Provides a connection option that allows you to configure the driver to report that it supports transactions, although Impala does not support transactions. This provides a workaround for applications that do not operate with a driver that reports that transcations are not supported.
    - Provides a connection option that allows you to set a default limit for the number of rows returned when an ORDER BY clause is submitted. This provides a workaround for applications that are not compatible with Impala's requirement that ORDER BY clauses limit the number of rows returned.
    - Supports the following standard SQL functionality:
      - Create Index and Create Table
      - Insert, Update, and Delete
      - Drop Index and Drop Table

See The Impala Wire Protocol Driver on page 384 for details.

Product Matrix

The DataDirect Connect Series *for* ODBC products include 32- and 64-bit drivers. DataDirect Connect *for* ODBC (32-bit) and DataDirect Connect64 *for* ODBC (64-bit) are detailed in the following table.

Driver	Connect *for* ODBC	Connect64 *for* ODBC
Progress OpenEdge® Wire Protocol	X	X
Sybase Wire Protocol	X	X
Text	X	X
Btrieve	X	
dBASE	X	
XML	X	

DataDirect Connect XE *for* ODBC (32-bit) and DataDirect Connect64 XE *for* ODBC (64-bit) products consists of the drivers detailed in the following table.

Driver	Connect XE *for* ODBC	Connect64 XE *for* ODBC
Greenplum Wire Protocol	X	X
Impala™ Wire Protocol	X	X
Driver for the Teradata Database	X	X

Product Platforms

For the latest information on the platforms supported by DataDirect Connect Series *for* ODBC drivers, refer to DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

About the Documentation Library

The documentation library is available on the Progress DataDirect Connectors Documentation Hub: https://docs.progress.com/bundle/datadirect-connectors/page/DataDirect-Connectors-by-data-source.html.

The DataDirect Connect Series for ODBC includes the following documents:

- *DataDirect Connect Series for ODBC Installation Guide* details requirements and procedures for installing the product.

- *DataDirect Connect Series for ODBC User's Guide* provides information about configuring and using the product.
- *DataDirect Connect Series for ODBC Reference* provides detailed reference information about the product.
- *DataDirect Connect Series for ODBC Troubleshooting Guide* provides information about error messages and troubleshooting procedures for the product.

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.

2

Quick Start Connect

This chapter provides basic information about configuring your driver immediately after installation and testing your connection. To take full advantage of the features of the driver, read About the Product on page 37 and the driver-specific chapter.

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

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

For details, see the following topics:

- Configuring and Connecting on Windows
- Configuring and Connecting on UNIX and Linux

Configuring and Connecting on Windows

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

Configuring a Data Source

To configure a data source:

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

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

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

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

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

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

   ---

2. The following two options appear on the General tab of all driver Setup dialog boxes:

   Data Source Name: Type a string that identifies this data source configuration, such as Accounting.

   Description: Type an optional long description of a data source name, such as My Accounting Database.

   Provide the requested information for all other options on the General tab; then, click **Apply** to configure the data source.

Minimum Configuration Requirements (Windows)

The following section describes the minimum options required to establish a connection for drivers that support 32 and 64-bit platforms. For a complete list of supported connection options, refer to the chapter for your driver.

Greenplum Wire Protocol

Progress OpenEdge

Sybase Wire Protocol

Teradata

Greenplum Wire Protocol

Provide the following information on the General Tab:

- **Host Name**: Type either the name or the IP address of the server to which you want to connect.
- **Port Number**: Type the port number of the server listener. The default is `5432`.
- **Database Name**: Type the name of the database to which you want to connect by default.

Impala Wire Protocol

Provide the following information on the General Tab:

- **Host Name**: Type either the name or the IP address of the server to which you want to connect.
- **Port Number**: Type the port number of the server listener. The default is `21050`.
- **Database Name**: Type the name of the database to which you want to connect by default.

Progress OpenEdge Wire Protocol

Provide the following information on the General Tab:

- **Host Name**: Type either the name or the IP address of the server to which you want to connect.
- **Port Number**: Type the port number of the server listener.
- **Database Name**: Type the name of the database to which you want to connect by default.
- **User ID**: Type your user name of as specified on the Progress OpenEdge server.

Sybase Wire Protocol

Provide the following information on the General Tab:

- **Network Address**: Type the IP address of the server to which you want to connect. Specify this address as *IP_address*, *port_number*. For example, you can enter `199.226.224.34, 5000`.

  If your network supports named servers, you can specify an address as *server_name, port_number*. For example, you can enter `SybSserver, 5000`.

- **Database Name**: Type the name of the database to which you want to connect by default. If you do not specify a value, the default is the database defined by the system administrator for each user.

Driver for the Teradata Database

Following is a list of connection options on the General Tab:

- **DBCName** or **Alias**: Type the IP address or the alias name of the Teradata Server. Using an IP address reduces the time it takes to connect, but if that address is not available at connection time, the connection fails and the driver does not attempt to fail over to another address.

  Using an alias name increases the time it takes to connect because the driver must search a local hosts file to resolve the name to an IP address, but it allows the driver to try and connect to alternate IP addresses if the first address fails. If you use an alias name, you must have or create a local hosts file that contains the alias names. The alias name cannot be more than eight characters long.

- **DBCName List**: Type the IP addresses or the alias names that are to appear in the drop-down list of the logon dialog box. Separate the names with commas. The same restrictions apply as described for the DBCName or Alias option.

  Using an alias name increases the time it takes to connect because the driver must search a local hosts file to resolve the name to an IP address, but it allows the driver to try and connect to alternate IP addresses if the first address fails. If you use an alias name, you must have or create a local hosts file that contains the alias names. The alias name cannot be more than eight characters long.

- **Integrated Security**: Select this check box to enable the user to connect to the database through single sign-on (SSO) using one of the authentication mechanisms that support SSO. When this check box is not selected (the default), UserID is required.
- **Security Mechanism**: Select TD2 from the drop-down list to specify the authentication mechanism used for connections to the data source. Valid values are:
  - **Default**—uses TD2.
  - **KRB5**— uses Kerberos as the authentication mechanism on Windows clients working with Windows servers if the server is V2R6.0.
  - **KRB5C**— uses Kerberos Compatibility as the authentication mechanism on Windows clients working with Windows servers if the server is pre-V2R6.0.
  - **LDAP**—uses LDAP as the authentication mechanism.
  - **NTLM**— uses NTLM as the authentication mechanism on Windows clients working with Windows servers if the server is V2R6.0.: Type a string of characters that is to be regarded as a parameter to the authentication mechanism. The string is ignored by the ODBC driver and is passed on to the TeraSSO function that is called to set the authentication mechanism. The characters [] {} () , ; ? * = ! @ must be enclosed in curly braces.
  - **NTLMC**— uses NTLM Compatibility as the authentication mechanism on Windows clients working with Windows servers if the server is pre-V2R6.0.
  - **TD1**—uses Teradata 1 as the authentication mechanism.
  - **TD2**—(default) uses Teradata 2 as the authentication mechanism.
- **Security Parameter**
- **UserID**: Type the default UserID for the Teradata database.

Testing the Connection

To test the connection:

1. After you have configured the data source, you can click **Test Connect** on the Setup dialog box to attempt to connect to the data source using the connection options specified in the dialog box. Some drivers immediately return a message indicating success or failure. For most drivers, a logon dialog box appears as described in each individual driver chapter.
2. Supply the requested information in the logon dialog box and click **OK**. Note that the information you enter in the logon dialog box during a test connect is not saved.
   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.
3. On the driver Setup dialog box, click **OK**. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the previously described procedure to modify your data source. You can override these defaults by connecting to the data source using a connection string with alternate values. See individual driver chapters for information about using connection strings.

Configuring and Connecting on UNIX and Linux

UNIX®

The following basic information enables you to configure a data source and test connect with a driver immediately after installation. See Configuring the Product on UNIX/Linux on page 87 for detailed information about configuring the UNIX/Linux environment and data sources.

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

Environment Configuration

To configure the environment:

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

   ```
   echo $SHELL
   ```

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

   ```
   env
   ```

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

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

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

   From the C shell, you would enter:

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

Test Loading the Driver

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

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

```
ivtestlib /opt/odbc/lib/ivase<i>xx</i>.so
```

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

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

Configuring a Data Source

In the 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 (see ODBCINI on page 88 for details about relocating and renaming this file). It is a plain text file that contains data source definitions. For instructions on configuring this file, see Data Source Configuration Through the System Information (odbc.ini) File on page 90.

Minimum Configuration Requirements (UNIX/Linux)

The following section describes the minimum options required to establish a connection for drivers that support both 32 and 64-bit platforms. For a complete list of supported connection options, refer to the chapter for your driver.

Greenplum Wire Protocol

Progress OpenEdge

Sybase Wire Protocol

Teradata

Greenplum Wire Protocol

The following example demonstrates the minimum connection information required to establish a connection:

```
[Greenplum Wire Protocol]
...
Driver=ODBCHOME/lib/xxgplmnn.zz
...
Database=Gplumdb1
...
HostName=GreenplumServer
...
PortNumber=5432
...
```

Connection option descriptions:

- `Database`: The name of the database to which you want to connect by default.
- `Hostname`: Either the name or the IP address of the server to which you want to connect.
- `PortNumber`: The port number of the server listener. The default is `5432`.

Impala Wire Protocol

The following example demonstrates the minimum connection information required to establish a connection:

```
[Impala Wire Protocol]
Driver=ODBCHOME/lib/xximpala.zz
...
Database=Impala1
...
HostName=ImpalaServer
...
PortNumber=21050
...
```

Connection option descriptions:

- `Database`: The name of the database to which you want to connect by default.
- `Hostname`: Either the name or the IP address of the server to which you want to connect.
- `PortNumber`: The port number of the server listener. The default is `21050`.

Progress OpenEdge Wire Protocol

The following example demonstrates the minimum connection information required to establish a connection:

```
[Progress OpenEdge Wire Protocol]
Driver=ODBCHOME/lib/xxoenn.zz
...
Database=odb1
...
HostName=OpenEdgeServer
...
PortNumber=5432
...
```

Connection option descriptions:

- `Database`: The name of the database to which you want to connect by default.
- `Hostname`: Either the name or the IP address of the server to which you want to connect.
- `PortNumber`: The port number of the server listener.

Sybase Wire Protocol

The following example demonstrates the minimum connection information required to establish a connection:

```
[Sybase Wire Protocol]
Driver=ODBCHOME/lib/xxasenn.zz
...
Database=master
...
NetworkAddress=123.226.224.34,5000
...
```

Connection option descriptions:

- `Database`: The name of the database to which you want to connect by default. If you do not specify a value, the default is the database defined by the system administrator for each user.
- `NetworkAddress`: The IP address of the server to which you want to connect. Specify this address as: *`IP_address, port_number`*. For example, you can enter `123.226.224.34, 5000`.

  If your network supports named servers, you can specify an address as: *`server_name`*, *`port_number`*. For example, you can enter `SSserver, 5000`.

Driver for the Teradata Database

Prerequisites:

- You must have all components of your database client software installed and connecting properly; otherwise, the driver will not operate correctly.

The following example demonstrates the minimum connection information required to establish a connection:
:

```
[Teradata]
Driver=ODBCHOME/lib/xxterann.zz
...
DBCName=123.123.12.12
...
SecurityMechanism=TD2
...
SecurityParameter=5678
...
UserID=John
...
```

Connection option descriptions:

- `DBCName`: The IP address or the alias name of the Teradata Server. Using an IP address reduces the time it takes to connect, but if that address is not available at connection time, the connection fails and the driver does not attempt to fail over to another address.

  Using an alias name increases the time it takes to connect because the driver must search a local hosts file to resolve the name to the IP address information, but it allows the driver to try and connect to alternate IP addresses if the first address fails. If you use an alias name, you must have or create a local hosts file that contains the alias names. The alias name cannot be more than eight characters long.
- `SecurityMechanism`: Enter TD2.
- `SecurityParameter`: A string of characters that is to be regarded as a parameter to the authentication mechanism. The string is ignored by the ODBC driver and is passed on to the TeraSSO function that is called to set the authentication mechanism. The characters `[] {} () , ; ? * = ! @` must be enclosed in curly braces.
- `UserID`: The default UserID for the Teradata database.

Testing the Connection

To test the connection (GUI environment):

1. After you have configured the data source, you can click **Test Connect** on the Setup dialog box to attempt to connect to the data source using the connection options specified in the dialog box. Some drivers immediately return a message indicating success or failure. For most drivers, a logon dialog box appears as described in each individual driver chapter.
2. Supply the requested information in the logon dialog box and click **OK**. Note that the information you enter in the logon dialog box during a test connect is not saved.
   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.
3. On the driver Setup dialog box, click **OK**. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the previously described procedure to modify your data source. You can override these defaults by connecting to the data source using a connection string with alternate values. See individual driver chapters for information about using connection strings.

3

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 (UNIX/Linux)."

For details, see the following topics:

- The Example application
- Microsoft Excel (Windows only)

The Example application

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

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

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

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

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

Note:

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

To use the Example application:

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

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

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

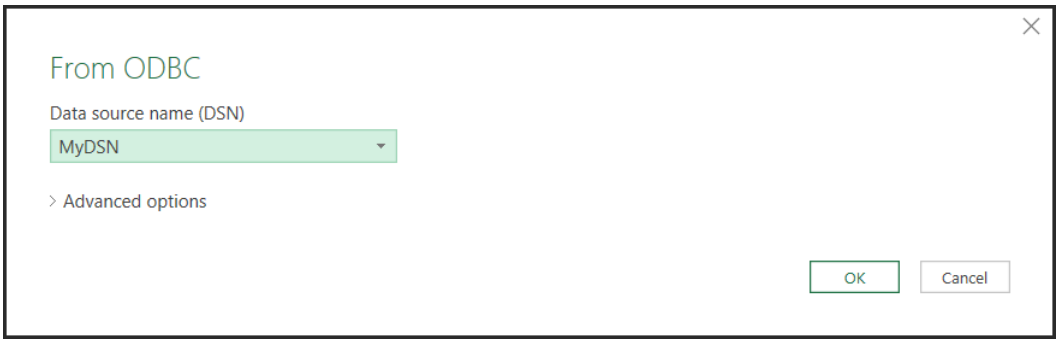
The results of your query are displayed. If example is unable to connect, the appropriate error message is returned.

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.

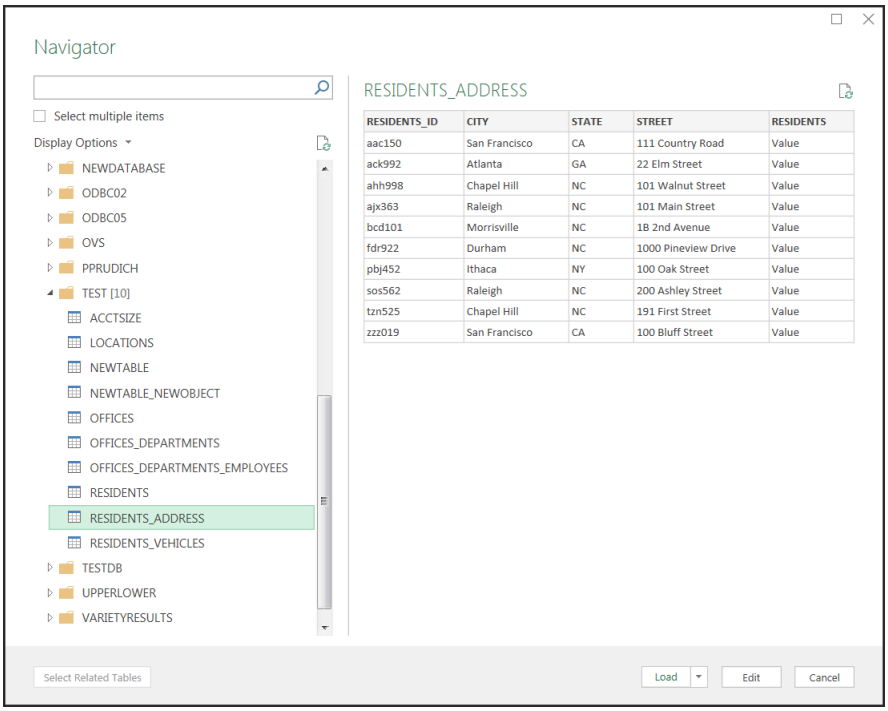
To use the driver to access data with Excel from the Data Connection Wizard:

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.

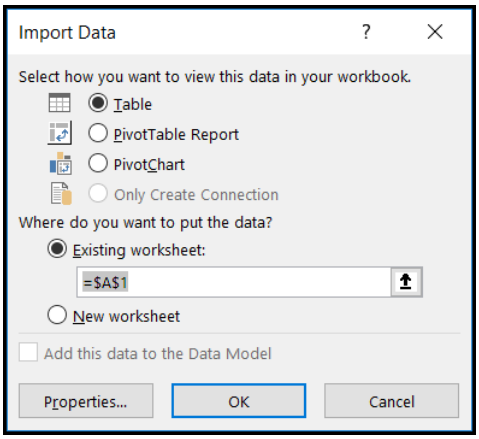


From the list, select the tables you want to access. A preview of your data will appear in the pane on the right. Optionally, click **Edit** to modify the results using the Query Editor. Refer to the Microsoft Excel product documentation for detailed information on using the Query Editor.

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/.

4

General Information on Using Connect Drivers

For details, see the following topics:

About the Product

The DataDirect Connect Series *for* ODBC drivers are compliant with the Open Database Connectivity (ODBC) specification and compatible with ODBC 3.8 applications.

Progress DataDirect provides ODBC drivers for both relational and flat-file database systems. The flat-file drivers provide full SQL support; see SQL Statements for Flat-File Drivers for details.

Support for Multiple Environments

Progress DataDirect provides ODBC-compliant database drivers for Windows, UNIX, and Linux operating systems. See Environment-Specific Information for an explanation of the environment-specific differences when using the database drivers in your operating environment.

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

Environment-Specific Information

The sections For Windows Users on page 38 and For UNIX and Linux Users on page 39 contain information specific to your operating environment.

The following sections refer to threading models.

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

For Windows Users

For the latest information on the platforms supported by DataDirect Connect Series *for* ODBC drivers, refer to DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

The following are requirements for the 32- and 64-bit drivers on Windows operating systems.

32-Bit Drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit drivers. If your application was built with 64-bit system libraries, you must use 64-bit drivers (see 64-Bit Drivers on page 38). The database to which you are connecting can be either 32-bit or 64-bit enabled.
- The following processors are supported:
  - x86: Intel
  - x64: Intel and AMD
- An application that is compatible with components that were built using Microsoft Visual Studio 2010 compiler and the standard Win32 threading model.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

64-Bit Drivers

- All required network software that is supplied by your database system vendors must be 64-bit compliant.

- The following processors are supported:
  - Intel
  - AMD
- An application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 14.00.40310.41 and the standard Windows 64 threading model.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Setup of the Drivers

The driver must be configured before it can be used. See Quick Start Connect on page 23 for information about using the Windows ODBC Administrator. See the individual driver chapters for details about driver configuration.

Driver Names

The prefix for all 32-bit driver file names is IV. The prefix for all 64-bit driver file names is DD. The file extension is .DLL, which indicates dynamic link libraries. For example, the 32-bit Progress OpenEdge Wire Protocol driver file name is IVOE*nn*.DLL, where *nn* is the revision number of the driver.

Refer to the readme file shipped with the product for the file name of each driver.

For UNIX and Linux Users

For the latest information on the platforms supported by DataDirect Connect Series *for* ODBC drivers, refer to DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

UNIX® The following are requirements for the 32- and 64-bit drivers on UNIX/Linux operating systems.

32-Bit Drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit drivers. If your application was built with 64-bit system libraries, you must use 64-bit drivers (see 64-Bit Drivers on page 41). The database to which you are connecting can be either 32-bit or 64-bit enabled.

AIX

- 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

---

Note: TERADATA USERS: When compiling an application on AIX for use with the driver for the Teradata database, you must use the `-brtl` option. For example, use `cc -o pgm pgm.o –brtl -lodbc` or `ld -o pgm –brtl pgm.o -lodbc`

HP-UX

- 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).

  All of the standard 32-bit UNIX drivers are supported on HP PA-RISC.
- 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)

  Note: All of the standard 32-bit UNIX drivers are supported on HP PA-RISC.

  For IPF, the following DataDirect Connect *for* ODBC are supported:
  - Progress OpenEdge Wire Protocol
  - Sybase Wire Protocol

  The following DataDirect Connect XE *for* ODBC drivers are supported:
  - Greenplum
  - Impala Wire Protocol

Linux

- The following processors are supported:
  - x86: Intel
  - x64: Intel and AMD
- 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).

Oracle Solaris

- 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 v. 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

Note: All of the standard 32-bit UNIX drivers are supported on Solaris SPARC.

For x86, the following DataDirect Connect *for* ODBC driver is supported:

- Sybase Wire Protocol

The following DataDirect Connect XE *for* ODBC driver is supported:

- Impala Wire Protocol

64-Bit Drivers

All required network software that is supplied by your database system vendors must be 64-bit compliant.

- The library path environment variable is:
  - `LD_LIBRARY_PATH` on Linux, HP-UX Itanium, and Oracle Solaris
  - `LIBPATH` on AIX

AIX

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

HP-UX

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

The following drivers are supported on IPF:

DataDirect Connect64 *for* ODBC drivers	• Progress OpenEdge Wire Protocol • Sybase Wire Protocol • Text
DataDirect Connect64 XE *for* ODBC drivers	• Greenplum Wire Protocol • Impala Wire Protocol • Teradata

Linux

- The following processors are supported:
  - Intel Itanium II (IPF)
  - x64: Intel and AMD
- For Itanium II: an application compatible with components that were built using g++ GNU project C++ Compiler version 3.3.2 and the Linux native pthread threading model (Linuxthreads)
- For x64: 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)

Note: The Greenplum Wire Protocol driver is the only Connect XE driver supported on Linux Itanium. II.

Oracle Solaris

- 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 v. 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

All of the standard 32-bit UNIX drivers are supported on Solaris SPARC. For x64,The following drivers are supported for Oracle Solaris:

DataDirect Connect *for* ODBC drivers	• Sybase Wire Protocol
DataDirect Connect XE *for* ODBC drivers	• Greenplum Wire Protocol • Impala Wire Protocol

AIX

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and The demoodbc Application for details.

You must also include the correct compiler switches if you are building 64-bit binaries. For example, to build demoodbc, you would use:

```
xlC_r –DODBC64 -q64 -qlonglong -qlongdouble -qvftable -o demoodbc
-I../include demoodbc.c -L../lib -lc_r -lC_r -lodbc
```

HP-UX 11 aCC

The ODBC drivers require certain runtime library patches. The patch numbers are listed in the readme file for your product. HP-UX patches are publicly available from the HP Web site www.hp.com.

HP updates the patch database regularly; therefore, the patch numbers in the readme file may be superseded by newer versions. If you search for the specified patch on an HP site and receive a message that the patch has been superseded, download and install the replacement patch.

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and The demoodbc Application for details. You must also include the +DD64 compiler switch if you are building 64-bit binaries. For example, to build demoodbc, you would use:

```
aCC -Wl,+s +DD64 -DODBC64 -o demoodbc -I../include demoodbc.c -L../lib -lodbc
```

Linux

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and The demoodbc Application for details.

You must also include the correct compiler switches if you are building 64-bit binaries. For example, to build demoodbc, you would use:

```
g++ -o demoodbc -DODBC64 -I../include demoodbc.c -L../lib -lodbc -lodbcinst -lc
```

Oracle Solaris

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and The demoodbc Application for details.

You must also include the -xarch=v9 compiler switch if you are building 64-bit binaries. For example, to build demoodbc, you would use:

```
CC -mt –DODBC64 -xarch=v9 -o demoodbc -I../include demoodbc.c -L../lib -lodbc –lCrun
```

Setup of the Environment and the Drivers

On UNIX and Linux, several environment variables and the system information file must be configured before the drivers can be used. See Configuring and Connecting on UNIX and Linux for a brief description of these variables. See the individual driver chapters for details about driver configuration. See Configuring the Product on UNIX/Linux for complete information about using the drivers on UNIX and Linux.

Driver Names

The drivers are ODBC API-compliant dynamic link libraries, referred to in UNIX and Linux as shared objects. The prefix for all 32-bit driver file names is iv. The prefix for all 64-bit driver file names is dd. The driver file names are lowercase and the extension is .so, the standard form for a shared object. For example, the 32-bit Progress OpenEdge Wire Protocol driver file name is ivoe*nn*.so, where *nn* is the revision number of the driver. For drivers on HP-UX PA-RISC only, the extension is .sl, for example, ivoe*nn*.sl.

Refer to the readme file shipped with your DataDirect product for the file name of each driver.

Using IP Addresses

The drivers support Internet Protocol (IP) addresses in IPv4 and IPv6 format as shown in the following tables.

Table 1: IP Address Formats Supported by 32- and 64-bit DataDirect Connect *for* ODBC Drivers

Driver	IPv4	IPv6
Progress OpenEdge Wire Protocol	All supported versions	All supported versions
Sybase Wire Protocol	All supported versions	Sybase 12.5.2 and higher

Table 2: IP Address Formats Supported by 32- and 64-bit DataDirect Connect XE *for* ODBC Drivers

Driver	IPv4	IPv6
Greenplum Wire Protocol	All supported versions	All supported versions
Impala Wire Protocol	All supported versions	Not supported

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 Progress OpenEdge server is specified in IPv4 format:

```
DRIVER=DataDirect 7.1 Progress OpenEdge Wire Protocol;
HOST=123.456.78.90;PORT=2055;
DB=OEACCT;UID=JOHN;PWD=XYZZYYou
```

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

```
DRIVER=DataDirect 7.1 Progress OpenEdge Wire Protocol;
HOST=2001:DB8:0000:0000:8:800:200C:417A;PORT=2055;
DB=OEACCT;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 7.1 Progress OpenEdge Wire Protocol;
HOST=2001:DB8:0:0:8:800:200C:417A;PORT=2055;
DB=OEACCT;UID=JOHN;PWD=XYZZYYou
```

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

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

Binding Parameter Markers

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

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

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

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

To ensure interoperability, the DataDirect Connect *for* ODBC driver uses only the parameter binding information that is provided by the application. Some DBMSs cannot publish dynamic parameter information back to an ODBC driver. For example, Oracle databases can determine that a parameter is an integer; however, the Oracle query process cannot publish this information back to the driver.

Driver Threading Information

The following tables summarize the threading information available at this time for the drivers. Always consult the readme file for the most up-to-date information as threading information is subject to change with new database transport and server revisions. Currently, the XML driver is the only thread-impaired driver.

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

Table 3: Threading Information for DataDirect Connect *for* ODBC

Driver	Fully Threaded	Thread Per Connect
Btrieve	X	
dBASE	X	
Progress OpenEdge		X
Sybase Wire Protocol		X
Text	X	

Table 4: Threading Information for DataDirect Connect XE *for* ODBC

Driver	Fully Threaded	Thread Per Connect
Greenplum Wire Protocol		X
Impala Wire Protocol		X
Teradata	X	

Version String Information

All drivers, except the flat-file drivers, have a version string of the format:

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

or

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

All flat-file drivers have a version string of the format:

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

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 bas component.

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

CCCC is the build number of a flat-file driver's flt component.

For example:

```
07.16.0002 (b0001, u0002, F0001)
      |__|  |___|  |___|  |___|
     Driver  Bas    Utl    Flt
```


On Windows, you can check the version string through the properties of the driver DLL. Right-click the driver DLL and select **Properties**. The Properties dialog box appears. On the Version tab, click **File Version** in the Other version information list box.

You can always check the version string of a driver on Windows by looking at the About tab of the driver's Setup dialog.

UNIX® On UNIX and Linux, you can check the version string by using the test loading tool shipped with the product. This tool, ivtestlib for 32-bit drives and ddtestlib for 64-bit drivers, is located in *install_directory*/bin.

The syntax for the tool is:

```
ivtestlib shared_object
```

or

```
ddtestlib shared_object
```

For example, for the 32-bit Sybase Wire Protocol driver on Oracle Solaris:

```
ivtestlib ivase27.so
```

returns:

```
07.16.0001 (B0002, U0001)
```

Note that the Sybase Wire Protocol driver is not a flat-file driver; therefore, there is no flt component listed in the example.

For example, for the Driver Manager on Solaris:

```
ivtestlib libodbc.so
```

returns:

```
07.16.0001 (U0001)
```

For example, for the 64-bit Driver Manager on Solaris:

```
ddtestlib libodbc.so
```

returns:

```
07.16.0001 (U0001)
```

For example, for 32-bit ICU component on Solaris:

```
ivtestlib libivicu27.so
07.16.0001
```

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 each driver's shared object:

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

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

Retrieving Data Type Information

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

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

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

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

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

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

    SQLINTEGER       strlenTypeName,
                     strlenSqlDataType,
                     strlenColumnSize;
```

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

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

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

        do {

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

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

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

For information about how a database's data types map to the standard ODBC data types, see the appropriate driver chapter in this book.

Persisting a Result Set as an XML Data File

The DataDirect Connect Series *for* ODBC drivers allow 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
```

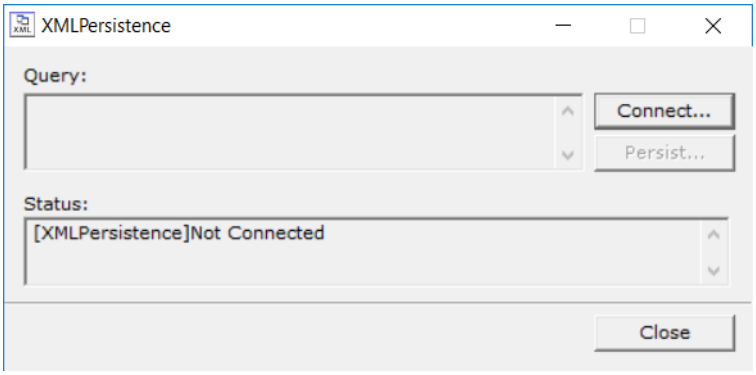
Using the Windows XML Persistence Demo Tool

The 32-bit drivers for Windows are shipped with an XML persistence demo tool. This tool is installed in the product installation directory.

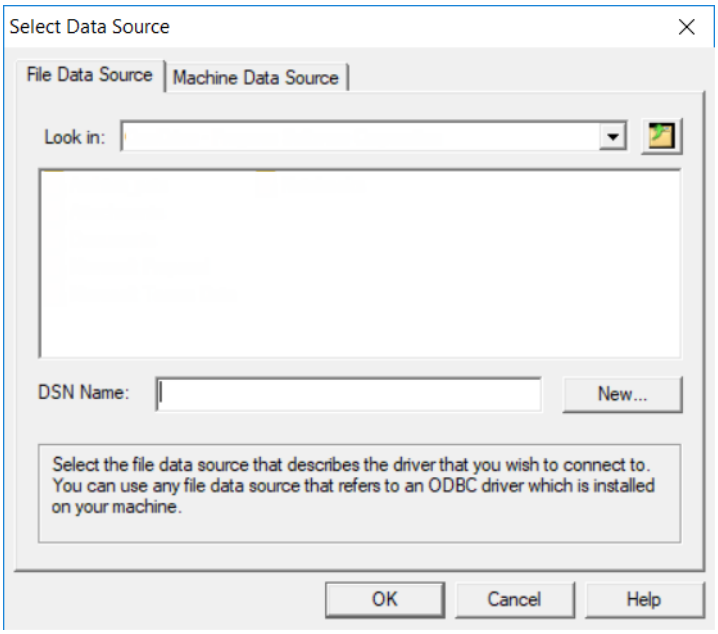
The tool has a graphical user interface and allows you to persist data as an XML data file.

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 drivers are 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.

Translators

Progress DataDirect provides a sample translator named "OEM to ANSI" that provides a framework for coding a translation library.

On Windows, refer to the readme.trn file in the \TRANSLAT subdirectory in the product installation directory.

UNIX® On UNIX and Linux, refer to the readme.trn file in the product installation directory, in the /samples/trn subdirectory.

Packet logging

The following 7.1.6 Connect for ODBC drivers support packet logging:

- Greenplum Wire Protocol
- Impala Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol
- Teradata

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

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

The following drivers support packet logging:

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

To enable TCP packet logging:

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

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

Configuring and enabling packet logging

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

Driver setup dialog (Windows)

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

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

odbc.ini file (UNIX/Linux)

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

The following example demonstrates enabling packet logging with the Sybase Wire Protocol driver:

```
[Sybase Wire Protocol]
Driver=ODBCHOME/lib/XXase28.so
Description=DataDirect 8.0 Sybase Wire Protocol
...
Database=master
...
EnablePacketLogging=1
...
NetworkAddress=123.226.224.34,5000
...
PacketLoggingFilePrefix=/tmp/myPacketLog
...
PacketLoggingMaxFileSize=102400
...
PacketLoggingMaxNumFiles=10
...
Password=secret
...
```

Connection string

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

The following demonstrates enabling packet logging with the SQL Server Wire protocol driver:

```
DRIVER={DataDirect 7.1 Sybase Wire Protocol};Database=master;
NetworkAddress=123.226.224.34,5000;EnablePacketLogging=1;
PacketLoggingFilePrefix=C:\temp\myPacketLog;
```

Packet logging connection options

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

Table 5: Packet Logging Connection Options

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

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

5

Advanced Features

The drivers support many advanced features, including:

- Using Failover on page 55
- Using DataDirect Connection Pooling on page 73
- Industry-standard security features such as Secure Socket Layer (SSL) data encryption and Kerberos authentication provide secure transmission of data. See Using Security on page 63 for more information.
- Using DataDirect Bulk Load on page 76

For details, see the following topics:

- Using Failover
- Using Security
- Using DataDirect Connection Pooling
- Using DataDirect Bulk Load
- Using Bulk Load for Batch Inserts

Using Failover

To ensure continuous, uninterrupted access to data, the DataDirect Connect Series *for* ODBC drivers provide the following levels of failover protection, listed from basic to more comprehensive:

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

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

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

- Connection Failover on page 56
- Extended Connection Failover on page 57
- Select Connection Failover on page 59

Regardless of the failover method you choose, you must configure one or multiple alternate servers using the Alternate Servers connection option. See Guidelines for Primary and Alternate Servers on page 60 for information about primary and alternate servers.

Connection Failover

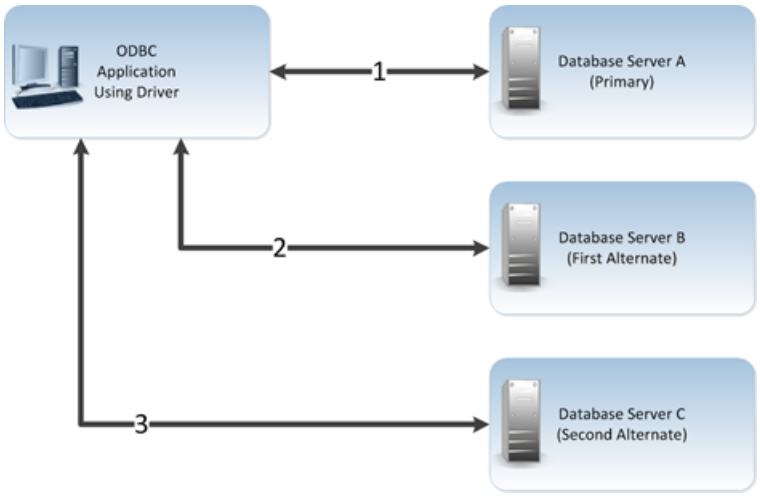
Connection failover is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol

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

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

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



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

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the driver can retry each alternate database server (primary and alternate) for a specified number of attempts. You can specify the number of attempts that are made through the *connection retry* feature. You can also specify the number of seconds of delay, if any, between attempts through the *connection delay* feature. See Using Connection Retry on page 60 for more information about connection retry.

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

For details on configuring connection failover for your driver, see the appropriate driver chapter in this book.

Extended Connection Failover

Extended connection failover is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol

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

- New connections, in the same way as described in Connection Failover on page 56
- Lost connections

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

- Restores the connection using the same connection options specified by the lost connection
- Reallocates statement handles and attributes
- Logs in the user to the database with the same user credentials
- Restores any prepared statements associated with the connection and repopulates the statement pool
- Restores manual commit mode if the connection was in manual commit mode at the time of the failover

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

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

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

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

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

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

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

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

Select Connection Failover

Select connection failover is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol

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

- New connections, in the same way as described in Connection Failover on page 56
- Lost connections, in the same way as described in Extended Connection Failover on page 57

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

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

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

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

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

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

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

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

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

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

Guidelines for Primary and Alternate Servers

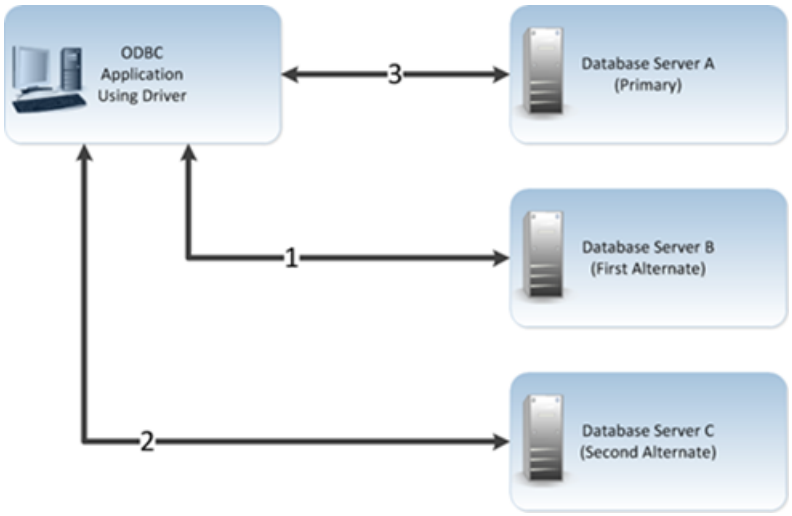
For failover to work correctly in your environment, ensure that the alternate servers mirror the data on the primary server or be part of a configuration where multiple database nodes share the same physical data.

Using Client Load Balancing

Client load balancing is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol

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



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

Client load balancing is controlled by the Load Balancing connection option. For details on configuring client load balancing, see the appropriate driver chapter in this book.

Using Connection Retry

Connection retry is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Progress OpenEdge Wire Protocol

- Sybase Wire Protocol

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

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

Using the connection options Connection Retry Count and Connection Retry Delay, you can specify the number of times the driver attempts to connect and the time in seconds between connection attempts. For details on configuring connection retry, see the appropriate driver chapter in this book.

Summary of Failover-Related Options

The following table summarizes how failover-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options. Not all options are available in every failover-enabled driver. The step numbers in the table refer the procedure that follows the table

Table 6: Summary: Failover and Related Connection Options

Option	Characteristic
Alternate Servers (See step 1 on page 62)	One or multiple alternate database servers. An IP address or server name identifying each server is required.
Connection Retry Count (See step 5 on page 62)	Number of times the driver retries the primary database server, and if specified, alternate servers until a successful connection is established.
Connection Retry Delay (See step 6 on page 62)	Wait interval, in seconds, between connection retry attempts when the Connection Retry Count option is set to a positive integer.
Failover Granularity (See step 3 on page 62)	The type of behavior that the driver exhibits when errors are detected during the failover process.
Failover Mode (See step 2 on page 62)	The type of failover that the driver attempts.
Failover Preconnect (See step 4 on page 62)	Determines whether the driver makes a connection attempt to the next server in the Alternate Servers list at the time of the initial connection.
Load Balancing (See step 7 on page 62)	Determines whether the driver uses client load balancing in its attempts to connect to primary and alternate database servers. If enabled, the driver attempts to connect to the database servers in random order.

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

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

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

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

A Connection String Example

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

```
DSN=AcctSybaseServer;AlternateServers=(NetworkAddress=123.456.78.90,5000:
Database=AccountingSybaseServer,NetworkAddress=098.765.43.21,5000:Database=Accounting);
ConnectionRetryCount=4;ConnectionRetryDelay=5;LoadBalancing=1;FailoverMode=0
```

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

An odbc.ini File Example

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

```
Driver=ODBCHOME/lib/ivasexx.so
Description=DataDirect Sybase Wire Protocol driver
...
AlternateServers=(NetworkAddress=123.456.78.90,5000:Database=AccountingSybaseServer,
NetworkAddress=098.765.43.21,5000:Database=Accounting)
...
ConnectionRetryCount=4
ConnectionRetryDelay=5
...
LoadBalancing=0
...
FailoverMode=1
...
FailoverPreconnect=1
...
```

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

Using Security

The drivers support the following security features:

- *Authentication* is the process of identifying a user.
- *Data encryption* is the conversion of data into a form that cannot be easily understood by unauthorized users.

Authentication

On most computer systems, a password is used to prove a user's identity. This password often is transmitted over the network and can possibly be intercepted by malicious hackers. Because this password is the one secret piece of information that identifies a user, anyone knowing a user's password can effectively be that user. Authentication methods protect the identity of the user.

The drivers support the following authentication methods:

- *User ID/password authentication* authenticates the user to the database using a database user name and password.
- *Client authentication* uses the user ID and password of the user logged onto the system on which the driver is running to authenticate the user to the database. The database server relies on the client to authenticate the user and does not provide additional authentication.
- *Kerberos authentication* is a trusted third-party authentication service that verifies user identities. DataDirect Connect Series *for* ODBC supports both Windows Active Directory Kerberos and MIT Kerberos implementations.
- *NTLM authentication* authenticates clients to the database through a challenge-response authentication mechanism that enables clients to prove their identities without sending a database password to the server.

Kerberos Authentication

Kerberos authentication is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Impala Wire Protocol
- Sybase Wire Protocol
- Driver for the Teradata Database

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

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

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

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

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

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

where *user* is the application user.

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

NTLM Authentication

The driver for the Teradata database supports NTLM (NTLMv1) authentication on Windows platforms.

Summary of Authentication-Related Options

The following table summarizes how authentication-related connection options work with the drivers. Note that some authentication-enabled drivers support only a subset of the listed options, as determined by natively supported features. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 7: Summary: Security Connection Options

Option	Characteristic
Authentication Method	The method the driver uses to authenticate the user to the server when a connection is established.

Option	Characteristic
GSS Client Library	The name of the Generic Security Service (GSS) client library that the driver uses to communicate with the Key Distribution Center (KDC).
Proxy User	Specifies the UserID used for impersonation.
Service Principal Name	The service principal name to be used by driver for Kerberos authentication.
User Name	The default user ID used to connect to your database.

Connection String Examples for Configuring Authentication

The following connection string configures the Sybase Wire Protocol driver to use authentication, specifically Kerberos authentication. The example contains the connection options necessary to configure Kerberos authentication as well as the minimum options required to establish a connection.

```
DSN=AcctSybaseServer;NetworkAddress=123.456.78.90,5000;
Database=AccountingSybaseServer;AuthenticationMethod=4;
GSSClient=native;ServicePrincipalName=myserver.example.com;
UID=JohnSmith;
```

odbc.ini File Examples for Configuring Authentication

The following example `odbc.ini` file configures the 32-bit Sybase Wire Protocol driver to use authentication, specifically Kerberos authentication. The example contains the connection options necessary to configure Kerberos authentication as well as the minimum options required to establish a connection.

```
Driver=ODBCHOME/lib/ivasexx.so
Description=DataDirect Sybase Wire Protocol driver
...
AuthenticationMethod=4
...
Database=AccountingSybaseServer
...
GSSClient=native
...
NetworkAddress=123.456.78.90,5000
...
ServicePrincipalName=myserver.example.com
...
UID=JohnSmith
...
```

Data Encryption Across the Network

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

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

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

Certain DataDirect Connect Series *for* ODBC drivers support Secure Sockets Layer (SSL). SSL is an industry-standard protocol for sending encrypted data over database connections. SSL secures the integrity of your data by encrypting information and providing client/server authentication.

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

TLS/SSL Encryption

TLS/SSL encryption is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Impala Wire Protocol
- Progress OpenEdge Wire Protocol
- Sybase Wire Protocol

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

- TLSv v1.0, v1.1, v1.2

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

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

The handshake involves the following types of authentication:

- *TLS/SSL server authentication* requires the server to authenticate itself to the client.
- *TLS/SSL client authentication* is optional and requires the client to authenticate itself to the server after the server has authenticated itself to the client. Not all databases support TLS/SSL client authentication.

Certificates

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

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

TLS/SSL Server Authentication

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

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

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

Additionally, the connection option, Host Name In Certificate (HostNameInCertificate), allows an additional method of server verification. When a value is specified for HostNameInCertificate, it must match the host name of the server, which has been established by the SSL administrator. This prevents malicious intervention between the client and the server and ensures that the driver is connecting to the server that was requested.

Finally, you can configure the driver to use the FIPS provider using the Enable FIPS (EnableFIPS) connection option. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2.

Note:

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

See also

Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 72

Designating an OpenSSL Library

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

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

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

File replacement

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

Windows:

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

UNIX/Linux:

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

Designating the absolute path to a library

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

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

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

TLS/SSL Client Authentication

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

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

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

Not all databases support TLS/SSL client authentication. The individual driver chapters indicate whether client authentication is supported.

For TLS/SSL client authentication, you can also configure the driver to load the FIPS provider using the Enable FIPS (EnableFIPS) connection option. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2.

Note:

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

See also

Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 72

Summary of Data Encryption Related Options

The following table summarizes how security-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 8: Summary: Security Connection Options

Option	Characteristic
Crypto Protocol Version	The cryptographic protocols the driver uses when SSL is enabled.
CryptoLibName	The absolute path for the OpenSSL library file containing the cryptographic library to be used by the data source or connection.
Encryption Method	The method the driver uses to encrypt data sent between the driver and the database server.
Enable FIPS	Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider.
Host Name In Certificate	The host name established by the SSL administrator for the driver to validate the host name contained in the certificate.
Key Password	The password required to access an individual key in the keystore.

Option	Characteristic
Keystore	The path that specifies the location of the keystore file.
Keystore Password	The password required to access the keystore.
PRNGSeedFile (UNIX/Linux only)	The absolute path for the entropy-source file or device used as a seed for SSL key generation.
PRNGSeedSource (UNIX/Linux only)	The source of the seed the driver uses for SSL key generation.
SSLLibName	The absolute path for the OpenSSL library file containing the SSL library to be used by the data source or connection.
Truststore	The path that specifies the location of the truststore file.
Truststore Password	The password required to access the truststore.
Validate Server Certificate	Validates the security certificate of the server as part of the SSL authentication handshake.

Connection String Examples for Configuring Data Encryption

The following connection strings configure the Greenplum Wire Protocol driver to use data encryption via the SSL server authentication and SSL client authentication methods. These examples contain the connection options necessary to configure data encryption as well as the minimum options required to establish a connection.

SSL Server Authentication

This connection string configures the driver to use the SSL Server Authentication method. In this configuration, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by the HostNameInCertificate option, and loads the FIPS provider for data encryption.

```
DSN=AcctGreenplumServer;EnableFIPS=1;EncryptionMethod=1;
HostName=GreenplumServer;HostNameInCertificate=MySubjectAltName;
PortNumber=5432;Truststore=TrustStoreLocation;Database=Accounting;
ValidateServerCertificate=1
```

SSL Client Authentication

This connection string configures the driver to use the SSL Server Authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate.

```
DSN=AcctGreenplumServer;EncryptionMethod=1;HostName=GreenplumServer;
HostNameInCertificate=MySubjectAltName;KeyPassword=YourKeyPassword;
Keystore=KeyStoreLocation;KeystorePassword=YourKSPassword;PortNumber=5432;
Database=Accounting;ValidateServerCertificate=1
```

Note: The OpenSSL 3.5 library and its providers are supported only with the Impala Wire Protocol Driver.

odbc.ini File Examples for Configuring Data Encryption

The following example `odbc.ini` files demonstrate how to configure the 64-bit Greenplum Wire Protocol driver to use data encryption via the SSL Server Authentication and SSL Client Authentication methods. These examples include the necessary options to configure data encryption as well as the minimum options required to establish a connection.

SSL Server Authentication

This `odbc.ini` file configures the driver to use the SSL Server Authentication method. In this configuration, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by the HostNameInCertificate option, and loads the FIPS provider for data encryption.

```
Driver=ODBCHOME/lib/ddgplmXX.so
Description=DataDirect Greenplum Wire Protocol driver
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=GreenplumServer
HostNameInCertificate=MySubjectAltName
...
PortNumber=5432
...
Database=Accounting
...
Truststore=TrustStoreLocation
...
ValidateServerCertificate=1
...
```

SSL Client Authentication

This `odbc.ini` file configures the driver to use the SSL Client Authentication method. In this configuration, since `ValidateServerCertificate=1`, the driver validates the certificate sent by the server and the host name specified by the HostNameInCertificate option.

```
Driver=ODBCHOME/lib/ddgplmXX.so
Description=DataDirect Greenplum Wire Protocol driver
...
EncryptionMethod=1
...
HostName=GreenplumServer
HostNameInCertificate=MySubjectAltName
...
KeyPassword=YourKeyPassword
Keystore=KeyStoreLocation
KeystorePassword=YourKSPassword
...
PortNumber=5432
...
Database=Accounting
...
ValidateServerCertificate=1
...
```

Note: The OpenSSL 3.5 library and its providers are supported only with the Impala Wire Protocol Driver.

Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms

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

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

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

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

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

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

Truststore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`truststore_filename`

is the name of the truststore file.

`cryptographic_algorithm`

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

`truststore_password`

is the password required for accessing the truststore file.

Example:

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

Keystore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`privatekey_file`

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

`truststore_filename`

is the name of the keystore file.

`cryptographic_algorithm`

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

Example:

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

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

Using DataDirect Connection Pooling

DataDirect connection pooling is available in the following DataDirect Connect Series *for* ODBC drivers:

- Greenplum Wire Protocol
- Sybase Wire Protocol

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

Note: Connection pooling works only with connections that are established using SQLConnect or SQLDriverConnect with the SQL_DRIVER_NO_PROMPT argument and only with applications that are thread-enabled.

DataDirect connection pooling that is implemented by the DataDirect driver is different than connection pooling implemented by the Windows Driver Manager. The Windows Driver Manager opens connections dynamically, up to the limits of memory and server resources. DataDirect connection pooling, however, allows you to control the number of connections in a pool through the Min Pool Size (minimum number of connections in a pool) and Max Pool Size (maximum number of connections in a pool) connection options. In addition, DataDirect connection pooling is cross-platform, allowing it to operate on UNIX and Linux. See the "Connection Option Descriptions" section in each driver's chapter for details about how the connection options manage DataDirect connection pooling.

Important: On a Windows system, do not use both Windows Driver Manager connection pooling and DataDirect connection pooling at the same time.

Creating a Connection Pool

Each connection pool is associated with a specific connection string. By default, the connection pool is created when the first connection with a unique connection string connects to the data source. The pool is populated with connections up to the minimum pool size before the first connection is returned. Additional connections can be added until the pool reaches the maximum pool size. If the Max Pool Size option is set to 10 and all connections are active, a request for an eleventh connection has to wait in queue for one of the 10 pool connections to become idle. The pool remains active until the process ends or the driver is unloaded.

If a new connection is opened and the connection string does not exactly match an existing pool, a new pool must be created. By using the same connection string, you can enhance the performance and scalability of your application.

Adding Connections to a Pool

A connection pool is created in the process of creating each unique connection string that an application uses. When a pool is created, it is populated with enough connections to satisfy the minimum pool size requirement, set by the Min Pool Size connection option. The maximum pool size is set by the Max Pool Size connection option. If an application needs more connections than the number set by Min Pool Size, The driver allocates additional connections to the pool until the number of connections reaches the value set by Max Pool Size.

Once the maximum pool size has been reached and no usable connection is available to satisfy a connection request, the request is queued in the driver. The driver waits for the length of time specified in the Login Timeout connection option for a usable connection to return to the application. If this time period expires and a connection has not become available, the driver returns an error to the application.

A connection is returned to the pool when the application calls SQLDisconnect. Your application is still responsible for freeing the handle, but this does not result in the database session ending.

Removing Connections from a Pool

A connection is removed from a connection pool when it exceeds its lifetime as determined by the Load Balance Timeout connection option. In addition, DataDirect has created connection attributes described in the following table to give your application the ability to reset connection pools. If connections are in use at the time of these calls, they are marked appropriately. When SQLDisconnect is called, the connections are discarded instead of being returned to the pool.

Table 9: Pool Reset Connection Attributes

Connection Attribute	Description
SQL_ATTR_CLEAR_POOLS Value: SQL_CLEAR_ALL_CONN_POOL	Calling SQLSetConnectAttr (SQL_ATTR_CLEAR_POOLS, SQL_CLEAR_ALL_CONN_POOL) clears all the connection pools associated with the driver that created the connection.This is a write-only connection attribute. The driver returns an error if SQLGetConnectAttr (SQL_ATTR_CLEAR_POOLS) is called.
SQL_ATTR_CLEAR_POOLS Value: SQL_CLEAR_CURRENT_CONN_POOL	Calling SQLSetConnectAttr (SQL_ATTR_CLEAR_POOLS, SQL_CLEAR_CURRENT_CONN_POOL) clears the connection pool that is associated with the current connection.This is a write-only connection attribute. The driver returns an error if SQLGetConnectAttr (SQL_ATTR_CLEAR_POOLS) is called.

Note: By default, if removing a connection causes the number of connections to drop below the number specified in the Min Pool Size option, a new connection is not created until an application needs one.

Handling Dead Connections in a Pool

What happens when an idle connection loses its physical connection to the database? For example, suppose the database server is rebooted or the network experiences a temporary interruption. An application that attempts to connect could receive errors because the physical connection to the database has been lost.

DataDirect Connect Series *for* ODBC drivers handle this situation transparently to the user. The application does not receive any errors on the connection attempt because the driver simply returns a connection from a connection pool. The first time the connection handle is used to execute a SQL statement, the driver detects that the physical connection to the server has been lost and attempts to reconnect to the server *before* executing the SQL statement. If the driver can reconnect to the server, the result of the SQL execution is returned to the application; no errors are returned to the application.

The driver uses connection failover option values, if they are enabled, when attempting this seamless reconnection; however, it attempts to reconnect even if these options are not enabled. See Connection Failover on page 56 for information about configuring the driver to connect to a backup server when the primary server is not available.

Note: If the driver cannot reconnect to the server (for example, because the server is still down), an error is returned indicating that the reconnect attempt failed, along with specifics about the reason the connection failed.

The technique that Progress DataDirect uses for handling dead connections in connection pools allows for maximum performance of the connection pooling mechanism. Some drivers periodically test the server with a dummy SQL statement while the connections sit idle. Other drivers test the server when the application requests the use of the connection from the connection pool. Both of these approaches add round trips to the database server and ultimately slow down the application during normal operation.

Connection Pool Statistics

Progress DataDirect has created a connection attribute to monitor the status of the DataDirect Connect Series *for* ODBC connection pools. This attribute, which is described in the following table, allows your application to fetch statistics for the pool to which a connection belongs.

Table 10: Pool Statistics Connection Attribute

Connection Attribute	Description
SQL_ATTR_POOL_INFO Value: SQL_GET_POOL_INFO	Calling SQLGetConnectAttr (SQL_ATTR_POOL_INFO, SQL_GET_POOL_INFO) returns a PoolInfoStruct that contains the statistics for the connection pool to which this connection belongs. This PoolInfoStruct is defined in qesqlext.h. For example:`SQLGetConnectAttr(hdbc, SQL_ATTR_POOL_INFO, PoolInfoStruct *, SQL_LEN_BINARY_ATTR(PoolInfoStruct), &len);`This is a read-only connection attribute. The driver returns an error if SQLSetConnectAttr (SQL_ATTR_POOL_INFO) is called.

Summary of Pooling-Related Options

The following table summarizes how connection pooling-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 11: Summary: Connection Pooling Connection Options

Option	Characteristic
Connection Pooling	Enables connection pooling.
Connection Reset	Resets a connection that is removed from the connection pool to the initial configuration settings of the connection.
Load Balance Timeout	An integer value to specify the amount of time, in seconds, to keep connections open in a connection pool.
Max Pool Size	An integer value to specify the maximum number of connections within a single pool.
Min Pool Size	An integer value to specify the minimum number of connections that are opened and placed in a connection pool when it is created.

Using DataDirect Bulk Load

The Sybase Wire Protocol driver supports DataDirect Bulk Load, a feature that allows your application to send large numbers of rows of data to a database.

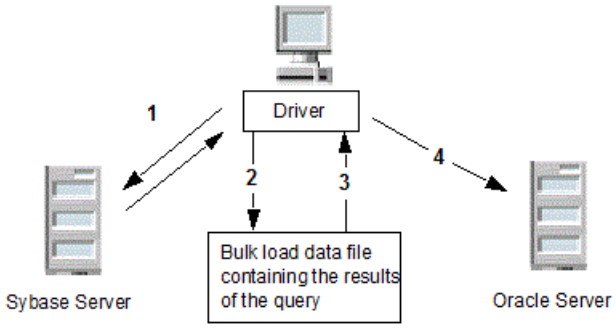
The driver sends the data to the database in a continuous stream instead of numerous smaller database packets. Similar to batch operations, using bulk load improves performance because far fewer network round trips are required. Bulk load bypasses the data parsing usually done by the database, providing an additional performance gain over batch operations.

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

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

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

Suppose that you had customer data on a Sybase server and need to export it to an Oracle server. The driver would perform the following steps:



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

Refer to "DataDirect Bulk Load" in the *Progress DataDirect for ODBC Drivers Reference* for supported functions and statement attributes.

Bulk Export and Load Methods

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

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

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

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

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

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

Exporting Data from a Database

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A bulk load configuration file with and .xml extension is also created when either a table or a result set is exported to a bulk load data file. See The Bulk Load Configuration File on page 81 for an example of a bulk load configuration file.

In addition, a log file of events as well as external overflow files can be created during a bulk export operation. The log file is configured through either the driver Setup dialog Bulk tab, the ExportTableToFile function, or the SQL_BULK_EXPORT statement attribute. The external overflow files are configured through connection options; see External Overflow Files on page 83 for details.

Bulk Loading to a Database

The Enable Bulk Load connection option specifies the method by which bulk data is loaded to a database. When the option is enabled, the driver uses database bulk load protocols. When not enabled, the driver uses standard parameter arrays.

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

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

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

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

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

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

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

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

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

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

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

Note: FOR SYBASE USERS: Additional database configuration is required for destination tables that do not have an index. See the "Persisting a Result Set as an XML Data File" section in your driver chapter for more information.

The Bulk Load Configuration File

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

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

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

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

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

For example, the format of the bulk load data file GBMAXTABLE.csv (discussed in Exporting Data from a Database on page 78) is defined by the bulk load configuration file, GBMAXTABLE.xml, as follows:

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

Bulk Load Configuration File Schema

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

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

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

Verification of the Bulk Load Configuration File

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

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

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

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

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

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

```
        printf ("Validate succeeded.\n");
}else {
    driverError (driverHandle, hmod);
}
```

Sample Applications

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

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

Character Set Conversions

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

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

Any code page listed in "Code page values" in the *Progress DataDirect for ODBC Drivers Reference* is supported for the bulk load data file.

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

The configuration file may optionally define a second code page value for each character column (`externalfilecodepage`). If character data is stored in an external overflow file (see External Overflow Files on page 83), this second code page value is used for the external file.

External Overflow Files

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

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

In the case of an external character data file, the character set of the file is governed by the bulk load configuration file. If the bulk load data file is Unicode and the maximum character size of the source data is 1, then the data is stored in its source code page. See Character Set Conversions on page 83.

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

Summary of Related options for DataDirect Bulk Load

The following table summarizes how DataDirect Bulk Load related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 12: Summary: Bulk Load Connection Options

Option	Characteristic
Batch Size	An integer value that specifies the number of rows at a time that the driver sends to the database during bulk operations.
Bulk Binary Threshold	An integer value that specifies the maximum size, in KB, of binary data exported to the bulk data file. Any data exceeding this size is exported to an external file.
Bulk Character Threshold	An integer value that specifies the maximum size, in KB, of character data exported to the bulk data file. Any data exceeding this size is exported to an external file.
Bulk Options	Toggles options for the bulk load process.
Field Delimiter	Specifies the character that the driver will use to delimit the field entries in a bulk load data file.
Record Delimiter	Specifies the character that the driver will use to delimit the record entries in a bulk load data file.

Using Bulk Load for Batch Inserts

For all drivers that support bulk operations, the driver uses the native bulk load protocol for database connections when the Enable Bulk Load connection option is set to `true`. For example, if you set the Enable Bulk Load connection option to `true`, the driver would use bulk load for the native parameter array insert request.

In some cases, the driver may not be able to use bulk load because of restrictions enforced by the bulk load protocol and will downgrade to a batch mechanism. For example, if the data being loaded has a data type that is not supported by the bulk load protocol, the driver cannot use bulk load, but will use the native parameter array insert mechanism instead.

For all drivers that support bulk operations, use the Bulk Load Batch Size connection option to specify the number of rows the driver loads at a time when bulk loading data. Performance can be improved by increasing the number of rows the driver loads at a time because fewer network round trips are required. Be aware that increasing the number of rows that are loaded also causes the driver to consume more memory on the client.

Determining the Bulk Load Protocol

Bulk operations can be performed using a dedicated bulk load protocol, that is, the protocol of the underlying database system, or by using parameter array batch operations. Dedicated protocols are generally more performance-efficient than parameter arrays. In some cases, however, you must use parameter arrays, for example, when the data to be loaded is in a data type not supported by the dedicated bulk protocol.

The Enable Bulk Load connection option determines bulk load behavior. When the option is enabled, the driver uses database bulk load protocols unless it encounters a problem, in which case it returns an error. In this situation, you must disable Enable Bulk Load so that the driver uses standard parameter arrays.

Summary of Related Options for Bulk Load or Batch Inserts

The following table summarizes how connection options related to bulk load for batch inserts work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 13: Summary: Bulk Load Connection Options

Option	Characteristic
Batch Size	An integer value that specifies the number of rows at a time that the driver sends to the database during bulk operations.
Enable Bulk Load	When enabled, the driver uses database bulk load protocols. When not enabled, the driver uses standard parameter arrays.

6

Configuring the Product on UNIX/Linux

UNIX®

This chapter contains specific information about using the DataDirect Connect Series *for* ODBC drivers in the UNIX and Linux environments.

See Environment-Specific Information on page 38 for additional platform information.

For details, see the following topics:

Environment Variables

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

Library Search Path

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

```
echo $SHELL
```

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

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

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

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

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

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

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

```
env
```

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

Some of the client-based drivers must have additional environment variables set. Consult the driver requirements in each of the individual driver chapters for additional environment variable information.

ODBCINI

Setup installs in the product installation directory a default system information file, named odbc.ini, that contains data sources. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 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

Setup installs in the product installation directory a default file, named odbcinst.ini, for use with DSN-less connections. See DSN-less Connections on page 98 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 Data Source Configuration Through the System Information (odbc.ini) File on page 90 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.

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

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

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

```
ivtestlib /opt/odbc/lib/ivase<i>xx</i>.so
```

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

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

See Version String Information on page 45 for details about version strings.

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

Data Source 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 (see ODBCINI on page 88 for details about relocating and renaming this file). It 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.

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

```
Sybase Wire Protocol=DataDirect Sybase Wire Protocol.
```

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

The `[ODBC Data Sources]` section also includes data source definitions. The default odbc.ini contains a data source definition for each driver. Each data source definition begins with a data source name in square brackets, for example, `[Sybase Wire Protocol 2]`. The data source definitions contain connection string *attribute=value* pairs with default values. You can modify these values as appropriate for your system. Descriptions of these attributes are in each individual driver chapter. See Sample Default odbc.ini File on page 92 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 on page 100).

Note: This section is not included in the default odbc.ini file that is installed by the product installer. You can 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/ivtrc27.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

The IANAAppCodePage keyword defines the default value that all UNIX/Linux drivers use if individual data sources have not specified a different value. The default value is 4.

See the individual driver chapters, and refer to "Code page values" in the *Progress DataDirect for ODBC Drivers Reference* for details.

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 drivers 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 on page 98 for details.

ODBC tracing allows you to trace calls to ODBC drivers 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*.

Sample Default odbc.ini File

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

```
[ODBC Data Sources]
dBASE=DataDirect 7.1 dBASEFile(*.dbf)
FoxPro3=DataDirect 7.1 dBASEFile(*.dbf)
Greenplum Wire Protocol=DataDirect 7.1 Greenplum Wire Protocol
Impala Wire Protocol=DataDirect 7.1 Impala Wire Protocol
Progress OpenEdge Wire Protocol=DataDirect 7.1 Progress OpenEdge Wire Protocol
Sybase Wire Protocol=DataDirect 7.1 Sybase Wire Protocol
Teradata=DataDirect 7.1 Teradata
Text=DataDirect 7.1 TextFile(*.*)

[dBASE]
Driver=ODBCHOME/lib/ivdbf27.so
Description=DataDirect 7.1 dBASEFile(*.dbf)
ApplicationUsingThreads=1
CacheSize=4
CreateType=dBASE5
Database=ODBCHOME/demo
DataFileExtension=DBF
ExtensionCase=UPPER
FileOpenCache=0
IntlSort=0
LockCompatibility=dBASE
Locking=RECORD
UseLongNames=0
UseLongQualifiers=0

[FoxPro3]
Driver=ODBCHOME/lib/ivdbf27.so
Description=DataDirect 7.1 dBASEFile(*.dbf)
ApplicationUsingThreads=1
CacheSize=4
CreateType=FoxPro30
Database=ODBCHOME/demo
DataFileExtension=DBF
ExtensionCase=UPPER
FileOpenCache=0
IntlSort=0
LockCompatibility=Fox
Locking=RECORD
UseLongNames=0
UseLongQualifiers=0

[Greenplum Wire Protocol]
Driver=ODBCHOME/lib/ivgplm27.so
Description=DataDirect 7.1 Greenplum Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
AuthenticationMethod=0
BatchMechanism=1
ConnectionRetryCount=0
ConnectionRetryDelay=3
CryptoLibName=
```

```
CryptoProtocolVersion=TLSv1.2, TLSv1.1, TLSv1,SSLv3
Database=<database_name>
EnableDescribeParam=1
EnableFIPS=1
EnableKeysetCursors=0
EncryptionMethod=0
ExtendedColumnMetaData=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchRefCursors=1
FetchTSWTZasTimestamp=0
FetchTWFSasTime=0
GSSClient=native
HostName=<Greenplum_host>
HostNameInCertificate=
IANAAppCodePage=4
InitializationString=
KeepAlive=0
KeyPassword=
KeysetCursorOptions=0
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxCharSize=
MaxLongVarcharSize=
MaxPoolSize=100
MaxVarcharSize=
MinPoolSize=0
Password=
Pooling=0
PortNumber=<Greenplum_server_port>
QueryTimeout=0
ReportCodepageConversionErrors=0
ServicePrincipalName=
SSLLibName=
TransactionErrorBehavior=1
TrustStore=
TrustStorePassword=
UnboundedNumericPrecision=1000
UnboundedNumericScale=6
ValidateServerCertificate=1
XMLDescribeType=-10

[Impala Wire Protocol]
Driver=ODBCHOME/lib/ivimpala27.so
Description=DataDirect Impala Wire Protocol Driver
ArraySize=50000
AuthenticationMethod=0
BatchMechanism=2
CryptoProtocolVersion=TLSv1.2,TLSv1.1,TLSv1,SSLv3
CryptoLibName=
Database=<database_name>
DefaultLongDataBuffLen=1024
DefaultOrderByLimit=-1
EnableDescribeParam=0
EnableFIPS=1
EncryptionMethod=0
GSSClient=native
HostName=<host_name>
HostNameInCertificate=
KeyPassword=
Keystore=
KeystorePassword=
LoginTimeout=30
LogonID=
```

```
MaxVarcharSize=
PortNumber=<Impala_server_port>
ProxyUser=
RemoveColumnQualifiers=0
ServicePrincipalName=
SSLLibName=
StringDescribeType=-9
TransactionMode=0
Truststore=
TruststorePassword=
UseCurrentSchema=0
ValidateServerCertificate=1

[Progress OpenEdge Wire Protocol]
Driver=ODBCHOME/lib/ivoe27.so
Description=DataDirect 7.1 Progress OpenEdge Wire Protocol
AlternateServers=
ArraySize=
ConnectionRetryCount=0
ConnectionRetryDelay=3
CryptoLibName=
CryptoProtocolVersion=TLSv1.2, TLSv1.1, TLSv1,SSLv3
Database=<database_name>
DefaultIsolationLevel=1
EnableFIPS=1
EnableTimestampWithTimezone=1
Encryption Method=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
HostName=<Progress_server>
HostNameInCertificate=
KeepAlive=0
LoadBalancing=0
LoginTimeout=15
PortNumber=<Progress_server_port>
QueryTimeout=0
SSLLibName=
TrustStore=
TrustStorePassword=
UseWideCharacterTypes=0
ValidateServerCertificate=1

[Sybase Wire Protocol]
Driver=ODBCHOME/lib/ivase27.so
Description=DataDirect 7.1 Sybase Wire Protocol
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=50
AuthenticationMethod=0
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadFieldDelimiter=
BulkLoadRecordDelimiter=
Charset=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CryptoLibName=
CryptoProtocolVersion=TLSv1.2, TLSv1.1, TLSv1,SSLv3
CursorCacheSize=1
Database=<database_name>
DefaultLongDataBuffLen=1024
DistributedTransactionModel=0
EnableBulkLoad=0
EnableDescribeParam=0
EnableFIPS=1
```

```
EnableQuotedIdentifiers=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverNetworkAddress=
FailoverPreconnect=0
FetchTWFSasTime=1
GSSClient=native
HostNameInCertificate=
InitializationString=
InterfacesFileServerName=
Language=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
KeepAlive=0
MaxPoolSize=100
MinPoolSize=0
NetworkAddress=<Sybase_host,Sybase_server_port>
OptimizePrepare=1
PacketSize=0
Password=
Pooling=0
PRNGSeedFile=/dev/random
PRNGSeedSource=0
QueryTimeout=0
RaiserrorPositionBehavior=0
ReportCodePageConversionErrors=0
SelectMethod=0
ServicePrincipalName=
SSLLibName=
TightlyCoupledDistributedTransactions=
TruncateTimeTypeFractions=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
WorkStationID=
XAOpenStringParameters=

[Teradata]
Driver=ODBCHOME/lib/ivtera27.so
Description=DataDirect 7.1 Teradata
AccountString=
AuthenticationDomain=
AuthenticationPassword=
AuthenticationUserid=
CharacterSet=ASCII
Database=
DBCName=<Teradata_server>
EnableDataEncryption=0
EnableExtendedStmtInfo=0
EnableLOBs=1
EnableReconnect=0
IntegratedSecurity=0
LoginTimeout=20
LogonID=
MapCallEscapeToExec=0
MaxRespSize=8192
Password=
PortNumber=1025
PrintOption=N
ProcedureWithSplSource=Y
ReportCodePageConversionErrors=0
SecurityMechanism=
SecurityParameter=
ShowSelectableTables=1
TDProfile=
TDRole=
```

```
TDUserName=

[Text]
Driver=ODBCHOME/lib/ivtxt27.so
Description=DataDirect 7.1 TextFile(*.*)
AllowUpdateAndDelete=0
ApplicationUsingThreads=1
CacheSize=4
CenturyBoundary=20
Database=ODBCHOME/demo
DataFileExtension=TXT
DecimalSymbol=.
Delimiter=,
FileOpenCache=0
FirstLineNames=0
IntlSort=0
ScanRows=25
TableType=Comma
UndefinedTable=GUESS

[ODBC]
IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc27.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10

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

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

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

     Consult the "Connection String Attributes" table of each driver chapter for other specific attribute values.

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

  Important: The "Connection Option Descriptions" section of each driver chapter 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, `[Sybase Wire Protocol]`.
  d) Modify the attributes in the new definition as necessary based on your system specifics, for example, enter the host name and port number of your system in the appropriate location.

     Consult the "Connection String Attributes" table of each driver chapter for other specific attribute values.

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

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

Important: The "Connection String Attributes" table of each driver chapter lists both the long and short name of the attribute. When entering attribute names into odbc.ini, you must use the long name of the attribute. The short name is not valid in the odbc.ini file.

Translators

Progress DataDirect provides a sample translator named "OEM to ANSI" that provides a framework for coding a translation library. Refer to the readme.trn file in the /samples/src/trn subdirectory in the product installation directory for details.

To perform a translation with a particular driver, you must include the TranslationSharedLibrary keyword in that driver's data source definition in the odbc.ini file. The TranslationSharedLibrary keyword represents the full path to the translation library.

For example, the 32-bit Progress OpenEdge driver would be:

```
[Progress OpenEdge]
Driver=ODBCHOME/lib/ivoe27.so
Description=DataDirect 7.1 Progress OpenEdge Wire Protocol
TranslationSharedLibrary=ODBCHOME/lib/ivtrn27.so
```

The TranslationOption keyword is the ASCII representation of the 32-bit integer translation option. Use of the TranslationOption keyword is optional.

The demoodbc Application

Progress DataDirect ships an application, named demoodbc, that is installed in the /samples/demo subdirectory of the product installation directory. Once you have set up your environment and data source, use the demoodbc application to test your connection. The syntax to run the application is:

```
demoodbc -uid user_name -pwd passworddata_source_name
```

For example:

```
demoodbc -uid johndoe -pwd secret DataSource3
```

The demoodbc application is coded to execute a Select statement from a table named emp. If you have an emp table in your database, the results are returned. If you do not have an emp table, you receive the message: Invalid object name 'EMP'. This message confirms a successful connection to the database.

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

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 on page 89 for details about relocating and renaming this file). This is a plain text file that contains default DSN-less connection information. You should not normally need to edit this file. The content of this file is divided into several sections.

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

```
DataDirect Sybase 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 Data Source Configuration Through the System Information (odbc.ini) File on page 90 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.

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 32-bit drivers with file names beginning with iv; a 64-bit driver file would be identical except that driver names would begin with dd.

```
[ODBC Drivers]
DataDirect 7.1 dBASEFile (*.dbf)=Installed
DataDirect 7.1 Greenplum Wire Protocol=Installed
DataDirect 7.1 Impala Wire Protocol=Installed
DataDirect 7.1 Progress OpenEdge Wire Protocol=Installed
DataDirect 7.1 Sybase Wire Protocol=Installed
DataDirect 7.1 Teradata=Installed
DataDirect 7.1 TextFile (*.*)=Installed

[DataDirect 7.1 dBASEFile (*.dbf)]
Driver=ODBCHOME/lib/ivdbf27.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileExtns=*.dbf
FileUsage=1
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivdbf27.so
SQLLevel=0

[DataDirect 7.1 Greenplum Wire Protocol]
Driver=ODBCHOME/lib/ivgplm27.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
```

```
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivgplm27.so
SQLLevel=0

[DataDirect 7.1 Impala Wire Protocol]
Driver=ODBCHOME/lib/ivimpala27.so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivimpala27.so
SQLLevel=0

[DataDirect 7.1 Progress OpenEdge Wire Protocol]
Driver=ODBCHOME/lib/ivoe27.so
APILevel=1
ConnectFunctions=YYN
DriverODBCVer=3.52
SQLLevel=0

[DataDirect 7.1 Sybase Wire Protocol]
Driver=ODBCHOME/lib/ivase27.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivase27.so
SQLLevel=0

[DataDirect 7.1 Teradata]
Driver=ODBCHOME/lib/ivtera27.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivtera27s.so
SQLLevel=0

[DataDirect 7.1 TextFile (*.*)]
Driver=ODBCHOME/lib/ivtxt27.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileExtns=*.*
FileUsage=1
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivtxt27.so
SQLLevel=0

[Administrator]
HelpRootDirectory=ODBCHOME/adminhelp

[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/odbctrc27.so
ODBCTraceMaxFileSize=102400
ODBCTraceMaxNumFiles=10
```

File Data Sources

The Driver Manager on UNIX and Linux supports file data sources. The advantage of a file data source is that it can be stored on a server and accessed by other machines, either Windows, UNIX, or Linux. See Quick Start Connect on page 23 for a general description of ODBC data sources on both Windows and UNIX.

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

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

```
[ODBC]
Driver=DataDirect 7.1 Sybase Wire Protocol
NetworkAddress=123.456.78.90,5000
Database=AccountingSybaseServer
LogonID=JOHN
```

It must contain all basic connection information plus any optional attributes. Because it uses the "DRIVER=" keyword, an odbcinst.ini file containing the driver location must exist (see DSN-less Connections on page 98).

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\Sybasewp.dsn
```

or, on UNIX and Linux:

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

If no path is specified for the file data source, the Driver Manager uses the DefaultDSNDir property, which is defined in the `[ODBC File DSN]` setting in the odbc.ini file to locate file data sources (see Data Source Configuration Through the System Information (odbc.ini) File on page 90 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/Sybasewp.dsn;UID=james;PWD=test01
```

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.

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);
    ```

7

Drivers for 32-Bit and 64-Bit Platforms

The following sections describe the drivers that are available in both 32- and 64-bit versions. See Drivers Only Available for 32-Bit Platforms on page 239 and The Connect XE Drivers on page 327 for information on additional Connect Series drivers.

For details, see the following topics:

- The Progress OpenEdge Wire Protocol Driver
- The Sybase Wire Protocol Driver
- The Text Driver

The Progress OpenEdge Wire Protocol Driver

The DataDirect Connect *for* ODBC Progress OpenEdge® Wire Protocol driver (the Progress OpenEdge Wire Protocol driver) supports Progress OpenEdge database systems and services.

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

The Progress OpenEdge Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the Windows, UNIX, and Linux environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the Progress OpenEdge Wire Protocol driver.

Driver Requirements

There are no client requirements for the Progress OpenEdge Wire Protocol driver.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 109 and Connection Option Descriptions for OpenEdge Wire Protocol on page 111 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

UNIX®

On UNIX and Linux, data sources are configured and modified by editing the system information file (by default, odbc.ini) and storing default connection values there. See Configuring the Product on UNIX/Linux on page 87 for detailed information about the specific steps needed to set up the UNIX and Linux environments and to configure a data source.

Connection Option Descriptions for OpenEdge Wire Protocol on page 111 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (OpenEdge)


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

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

To configure a Progress OpenEdge data source:

1. Start the ODBC Administrator. On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect 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.

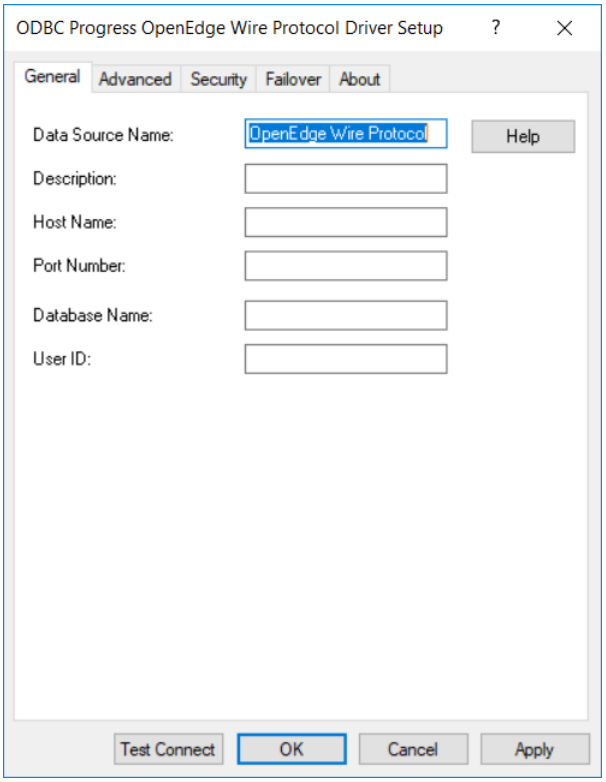
  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **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.

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

Figure 1: General tab



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

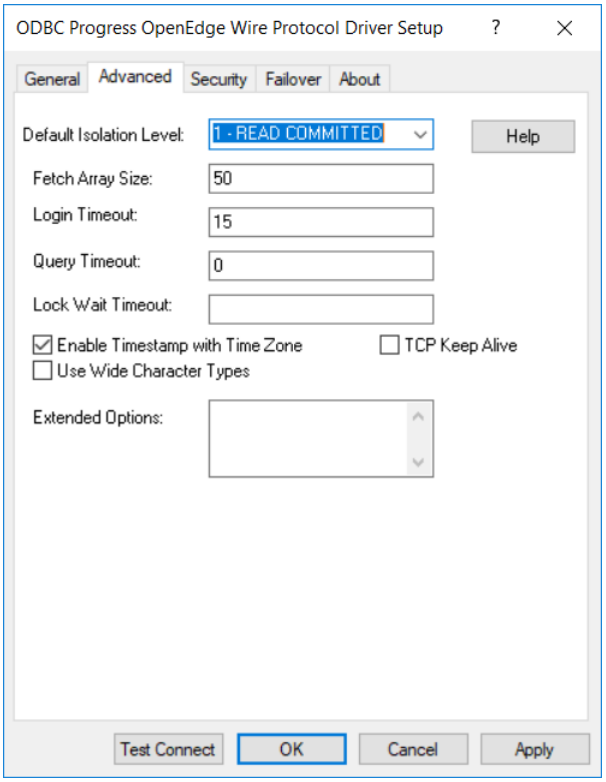
3. Provide values for the options on this tab in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 116	None
Description on page 118	None

Connection Options: General	Default
Host Name on page 123	None
Port Number on page 128	None
Database Name on page 117	None
User ID on page 133	None

4. Optionally, click the **Advanced** tab to specify additional data source settings.

Figure 2: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Default Isolation Level on page 117	1- READ COMMITTED
Fetch Array Size on page 123	50
Login Timeout on page 126	15
Query Timeout on page 129	0
Enable Timestamp with Timezone on page 119	Enabled
TCP Keep Alive on page 131	Disabled

Connection Options: Advanced	Default
Use Wide Character Types on page 133	Disabled
IANAAppCodePage on page 125 UNIX ONLY	4 (ISO 8559-1 Latin 1)

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

`Database=Server1;`UndocumentedOption1=*`value`*`[;`UndocumentedOption2=*`value`*`;]`

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

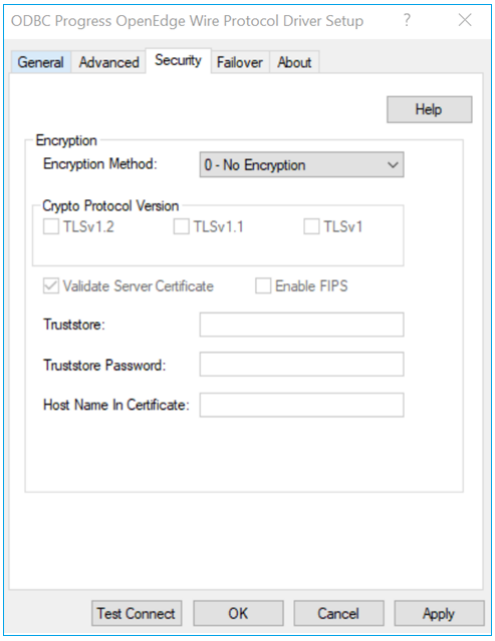
Note: Do not specify the Extended Options configuration option in a connection string, or the driver will return an error. Instead, applications should specify the individual undocumented connection options in the connection string.

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

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

5. Optionally, click the **Security** tab to specify security data source settings.

Figure 3: Security tab



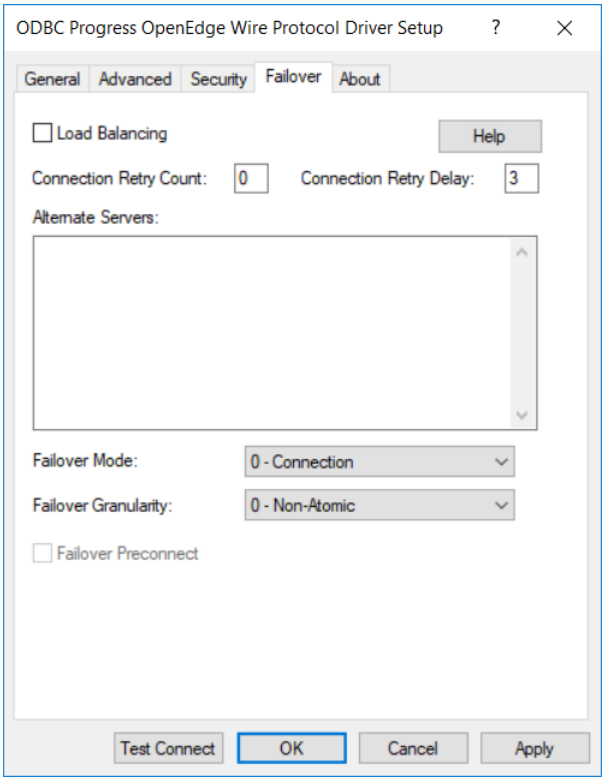
See Using Security on page 63 for a general description of authentication and encryption and their configuration requirements.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Security	Default
Encryption Method on page 120	0 (No Encryption)
Crypto Protocol Version on page 114	TLSv1.2, TLSv1.1, TLSv1, SSLv3
Validate Server Certificate on page 134	Enabled
Enable FIPS on page 119	Disabled
Truststore on page 131	None
Truststore Password on page 132	None
HostName In Certificate on page 124	None

6. Optionally, click the **Failover** tab to specify failover data source settings.

Figure 4: Failover tab



See Using Failover on page 55 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing on page 125	Disabled
Connection Retry Count on page 113	0
Connection Retry Delay on page 114	3
Alternate Servers on page 112	None
Failover Mode on page 121	0 (Connection)
Failover Granularity on page 120	0 (Non-Atomic)
Failover Preconnect on page 122	Disabled

7. 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 (OpenEdge) on page 110 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.
   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

   Note: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

8. Click **OK** or **Cancel**. If 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.

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 for OpenEdge Wire Protocol on page 111 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 Progress OpenEdge is:

```
DSN=PROGRESS;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

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

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

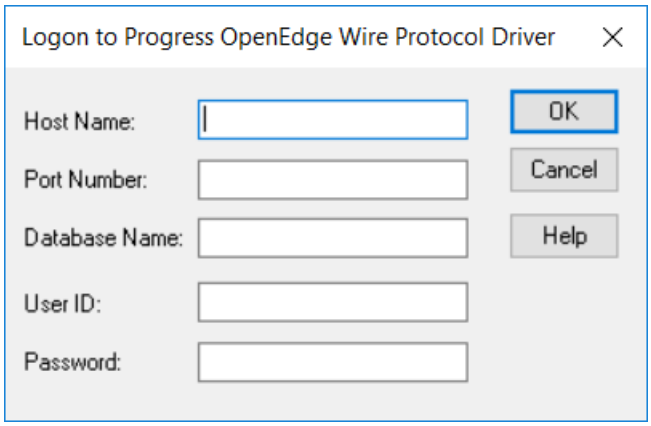
A DSN-less connection string must provide all necessary connection information:

```
DRIVER={DataDirect 7.1 Progress OpenEdge Wire Protocol};DB=PAYROLL;UID=JOHN;
PWD=XYZZY;HOST=LOCALHOST;PORT=2055
```

Using a Logon Dialog Box (OpenEdge)

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

Figure 5: Logon to Progress OpenEdge Wire Protocol Driver dialog box



In this dialog box, provide the following information:

1. In the Host Name field, type the name of the system where the database is stored.
2. Type the Port Number setup for the database listener process.
3. Type the name of the database to which you want to connect.
4. Type your user name.
5. Type your password.
6. Click **OK** to complete the logon.

Connection Option Descriptions for OpenEdge Wire Protocol

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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the Progress OpenEdge Wire Protocol driver.

Table 14: Progress OpenEdge Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ArraySize (AS)	None
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CryptoProtocolVersion (CPV)	TLSv1.2, TLSv1.1, TLSv1, SSLv3
CryptoLibName (CLN)	Empty string
Database (DB)	None
DataSourceName (DSN)	None
DefaultIsolationLevel (DIL)	1- READ COMMITTED
Description (n/a)	None
EnableFIPS (EF)	Disabled
EnableTimestampwithTimezone (ETWT)	1 (enabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
Fetch Array Size	50

Attribute (Short Name)	Default
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
KeepAlive (KA)	Disabled
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
OpenSSLConfigFile (OSSLCNF)	*install_dir*\drivers\openssl.cnf (Windows) *install_dir*/lib/openssl.cnf (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*install_dir*\drivers (Windows) *install_dir*/lib (UNIX/Linux)
Password (PWD)	None
PortNumber (PORT)	None
QueryTimeout (QT)	0 (Disabled)
SSLLibName (SLN)	Empty string
Truststore (TS)	None
TruststorePassword (TSP)	None
UseWideCharacterTypes (UWCT)	0 (disabled)
ValidateServerCertificate (VSC)	1 (enabled)

Alternate Servers

Attribute

AlternateServers (ASVR)

Purpose

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

Valid Values

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

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

Notes

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

Example

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

```
AlternateServers=(HostName=123.456.78.90:PortNumber=5177:Database=PAYROLL1,
HostName=223.456.78.90:PortNumber=5178:Database=PAYROLL2)
```

Default

None

GUI Tab

Failover tab

Connection Retry Count

Attribute

ConnectionRetryCount (CRC)

Purpose

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

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

Valid Values

0 | x

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

0

GUI Tab

Failover tab

Connection Retry Delay

Attribute

ConnectionRetryDelay (CRD)

Purpose

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

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

Valid Values

0 | *x*

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

3

GUI Tab

Failover tab

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.2 | TLSv1.3

Example

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

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

Default

TLSv1.2,TLSv1.1

GUI Tab

Security tab

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

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

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

Valid Values

absolute_path\openssl_filename

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

Empty string

GUI Tab

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

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

See Advanced tab for details.

See also

SSLLibName on page 130

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

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

Default Isolation Level

Attribute

DefaultIsolationLevel (DIL)

Purpose

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

Valid Values

0 | 1 | 2 | 3

Behavior

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

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

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

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

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

Valid Values

Default

1 - READ COMMITTED

GUI Tab

Advanced tab

Description

Attribute

Description (n/a)

Purpose

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

Valid Values

`string`

where:

`string`

is a description of a data source.

Default

None

GUI Tab

General tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

`0`

Enable Timestamp with Timezone

Attribute

EnableTimestampwithTimezone (ETWT)

Purpose

Determines whether the driver exposes timestamps with timezones to the application.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver exposes timestamps with timezones to the application.

If set to 0 (Disabled), timestamps with timezones are not exposed to the application.

Default

1 (Enabled)

GUI Tab

Advanced tab

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

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

Valid Values

0 | 1

Behavior

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS v1, SSL v3, or SSL v2 in that order.

Notes

- This connection option can affect performance.

Default

0 (No Encryption)

GUI Tab

Security tab

See Also

See Performance Considerations on page 135 for details.

Failover Granularity

Attribute

FailoverGranularity (FG)

Purpose

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

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

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

Valid Values

0 | 1 | 2 | 3

Behavior

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

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

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

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

Default

0 (Non-Atomic)

GUI Tab

Failover tab

Failover Mode

Attribute

FailoverMode (FM)

Purpose

Specifies the type of failover method the driver uses.

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

Notes

- This connection option can affect performance.

Default

0 (Connection)

GUI Tab

Failover tab

See Also

See Performance Considerations on page 135 for details.

Failover Preconnect

Attribute

FailoverPreconnect (FP)

Purpose

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

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

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

Valid Values

0 | 1

Behavior

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

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

Default

0 (Disabled)

GUI Tab

Failover tab

Fetch Array Size

Attribute

ArraySize (AS)

Purpose

The number of rows the driver retrieves from the server for a fetch. This is not the number of rows given to the user.

This connection option can affect performance. See Performance Considerations on page 135 for details.

Valid Values

x

where:

x

is a positive integer specifying the number of bytes.

Notes

- This connection option can affect performance.

Default

50

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 135 for details.

Host Name

Attribute

HostName (HOST)

Purpose

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

Valid Values

server_name | *IP_address*

where:

server_name

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

IP_address

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

The IP address can be specified in either IPv4 or IPv6 format. See Using IP Addresses on page 43 for details about these formats.

Default

None

GUI Tab

General tab

HostName In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

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

Valid Values

host_name | *#SERVERNAME#*

where:

host_name

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

Behavior

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

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

Default

None

GUI Tab

Security tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

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

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

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

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

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

Valid Values

IANA_code_page

where:

IANA_code_page

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

Notes

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

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

N/A

Load Balancing

Attribute

LoadBalancing (LB)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

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

Default

0 (Disabled)

GUI Tab

Failover tab

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

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

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that represents a number of seconds.

Behavior

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

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

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

Default

15

GUI Tab

Advanced tab

OpenSSLConfigFile

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

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

Valid Values

`fips_config_file`

where:

`fips_config_file`

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

Notes

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

Default

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

OpenSSLProviderPath

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

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

Valid Values

provider_path

where:

provider_path

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

Notes

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

Default

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

Password

Attribute

Password (PWD)

Purpose

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

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

The port number of the server listener.

Valid Values

port_name

where:

port_name

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

Default

None

GUI Tab

General tab

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

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

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

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

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

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

Default

0

GUI Tab

Advanced tab

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

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

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

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

No default value

GUI Tab

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

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

See Advanced tab for details.

See also

CryptoLibName on page 115

TCP Keep Alive

Attribute

KeepAlive (KA)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

0 (Disabled)

GUI Tab

Advanced tab

Truststore

Attribute

Truststore (TS)

Purpose

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

Valid Values

truststore_directory\filename

where:

truststore_directory

is the directory where the truststore file is located

filename

is the file name of the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

Truststore Password

Attribute

TruststorePassword (TSP)

Purpose

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

Valid Values

truststore_password

where:

truststore_password

is a valid password for the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

Use Wide Character Types

Attribute

UseWideCharacterTypes (UWCT)

Purpose

A value that determines whether character data types are described to the application as SQL_CHAR or SQL_WCHAR when connected to a Unicode database.

Valid Values

0 | 1

Behavior

If set to 0 (Disabled), character data types are described to the application as SQL_CHAR.

If set to 1 (Enabled), character data types are described to the application as SQL_WCHAR.

Default

0 (Disabled)

GUI Tab

Advanced tab

User ID

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

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

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

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI Tab

Security tab

Performance Considerations

The following connection options can enhance driver performance.

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

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

Fetch Array Size (ArraySize): Reducing the number of round trips on the network to the approximate number of rows being fetched increases performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network.

Notes

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

Data Types

The following table shows how the Progress OpenEdge data types are mapped to the standard ODBC data types.

Table 15: Progress OpenEdge Data Types

OpenEdge	ODBC
Bigint	SQL_BIGINT
Binary	SQL_BINARY
Bit	SQL_BIT
Blob	SQL_LONGVARBINARY
Char	SQL_CHAR
Clob	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double precision	SQL_DOUBLE
Float	SQL_FLOAT
Integer	SQL_INTEGER

OpenEdge	ODBC
Lvarbinary	SQL_LONGVARBINARY
Lvarchar	SQL_LONGVARCHAR
Numeric	SQL_NUMERIC
Real	SQL_FLOAT
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Timestamp with Time Zone	SQL_CHAR
Tinyint	SQL_TINYINT
Varbinary	SQL_VARBINARY
Varchar	SQL_VARCHAR

Unicode Support

When connected to a Unicode database, the Progress OpenEdge Wire Protocol driver supports the Unicode data types listed in the following table, in addition to standard ODBC data types listed in Data Types on page 135. The Use Wide Character Types connection string option must be enabled.

Progress OpenEdge Data Type	Mapped to. . .
Char	SQL_WCHAR
Varchar	SQL_WVARCHAR

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

See UTF-16 Applications on UNIX and Linux on page 101 for related details.

Also, refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for a more detailed explanation of Unicode.

Advanced Features

The driver supports the following advanced features:

- Failover
- Security

Failover

The driver supports failover and its related connection options. Failover connection options are located on the Failover tab of the driver Setup dialog box. See Using Failover on page 55 for a general description of failover and its implementation.

Security

The driver supports data security by providing both user authentication and SSL data encryption, including Kerberos authentication and SSL data encryption. See Using Security on page 63 for a general description of authentication and encryption and its configuration requirements.

You configure the driver for data security on the Security tab of the driver Setup dialog box. See the description of the Security tab under Configuring and Connecting to Data Sources on page 138 for specific implementations.

Isolation and Lock Levels Supported

Progress OpenEdge supports isolation level 0 (read uncommitted), isolation level 1 (read committed), isolation level 2 (repeatable read), and isolation level 3 (serializable).

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

SQL Grammar Support

The driver supports the core SQL grammar.

ODBC Conformance Level

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

The driver also supports the function SQLSetPos.

The driver supports the function SQLCancel for SELECT statements with OpenEdge V12.4 and higher. If a SELECT statement is cancelled during the first or a subsequent fetch operation, the driver returns a "query aborted" exception. This function can be used by a thread to cancel a statement that is being executed by another thread. One or more statements may be cancelled if the method is called on a statement object that is executing multiple statements simultaneously and the driver may not return expected results. The driver supports this function for ODBC 3.x applications only.

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

Number of Connections and Statements Supported

The Progress OpenEdge database system supports multiple connections and multiple statements per connection.

The Sybase Wire Protocol Driver

The DataDirect Connect *for* ODBC and DataDirect Connect64 *for* ODBCSybase Wire Protocol driver (the Sybase Wire Protocol driver) each support the following databases and services:

- SAP Adaptive Server Enterprise
- Sybase Adaptive Server Enterprise

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

The Sybase Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the Sybase Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 152 and Connection Option Descriptions for Sybase Wire Protocol on page 153 for an alphabetical list of driver connection string attributes and their initial default values

Data Source Configuration in the UNIX/Linux odbc.ini File

UNIX® On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration on page 27 for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Connection Option Descriptions for Sybase Wire Protocol on page 153 lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (Sybase)


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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

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

To configure a Sybase data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

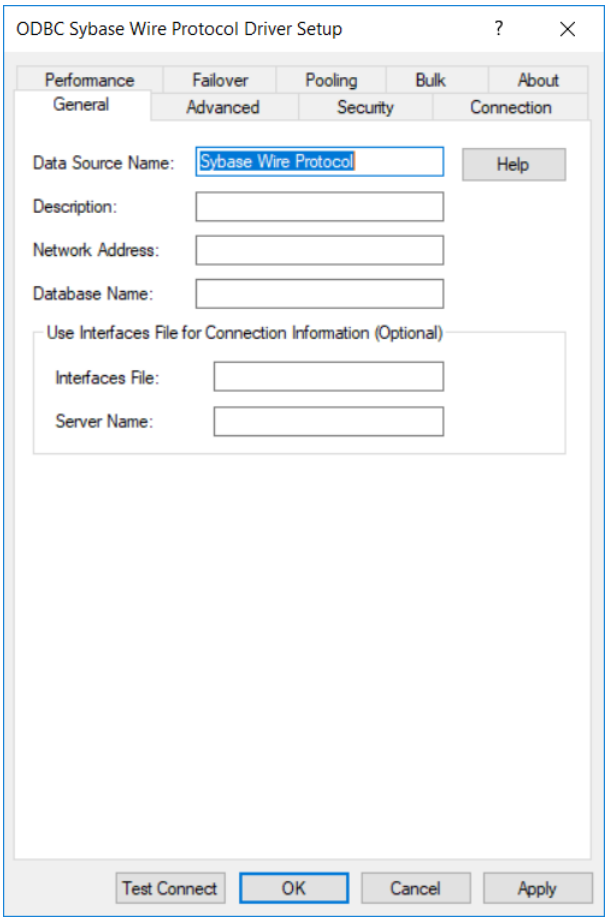
     If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

   - **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.

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

Figure 6: General tab



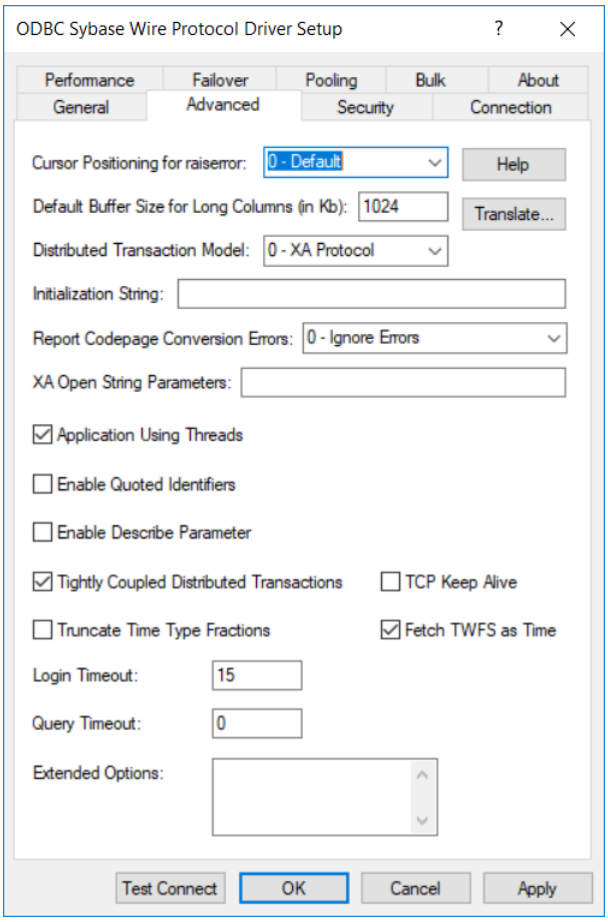
Note: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3. On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 168	None
Description on page 169	None
Network Address on page 187	None
Database Name on page 169	None
Interfaces File on page 182	None
Server Name on page 196	None

4. Optionally, click the **Advanced** tab to specify additional data source settings.

Figure 7: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Cursor Positioning for Raiserror on page 167	0 - Default
Default Buffer Size for Long/LOB Columns (in Kb) on page 170	1024
Distributed Transaction Model on page 171	0 - XA Protocol
Initialization String on page 182	None
Report Codepage Conversion Errors on page 195	0 - Ignore Errors
XA Open String Parameters on page 203	None
Application Using Threads on page 158	Enabled
Enable Quoted Identifiers on page 173	Disabled
Enable Describe Parameter on page 172	Disabled

Connection Options: Advanced	Default
Tightly Coupled Distributed Transactions on page 199	Enabled
TCP Keep Alive on page 199	Disabled
Truncate Time Type Fractions on page 200	Disabled
Fetch TWFS as Time on page 177	Enabled
Login Timeout on page 185	15
Query Timeout on page 194	0
IANAAppCodePage on page 181 UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

`Database=Server1;`UndocumentedOption1=*`value`*`[;`UndocumentedOption2=*`value`*`;]`

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

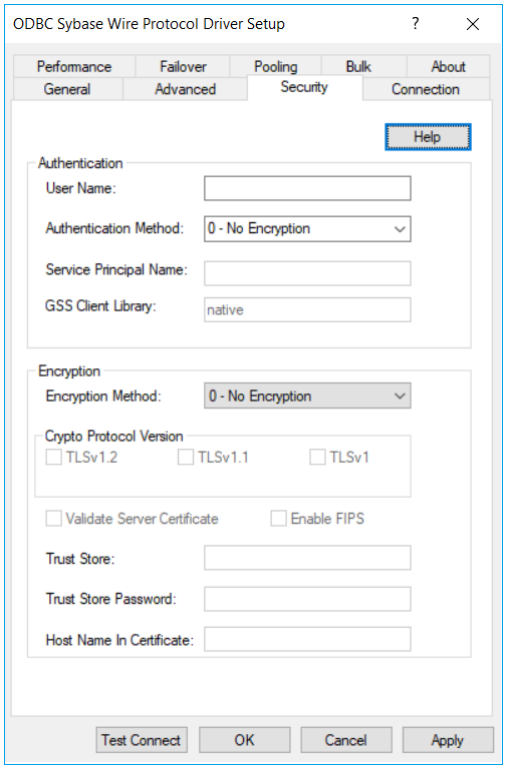
Note: Do not specify the Extended Options configuration option in a connection string, or the driver will return an error. Instead, applications should specify the individual undocumented connection options in the connection string.

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

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

5. Optionally, click the **Security** tab to specify security data source settings.

Figure 8: Security tab



See Using Security on page 63 for a general description of authentication and encryption and their configuration requirements.

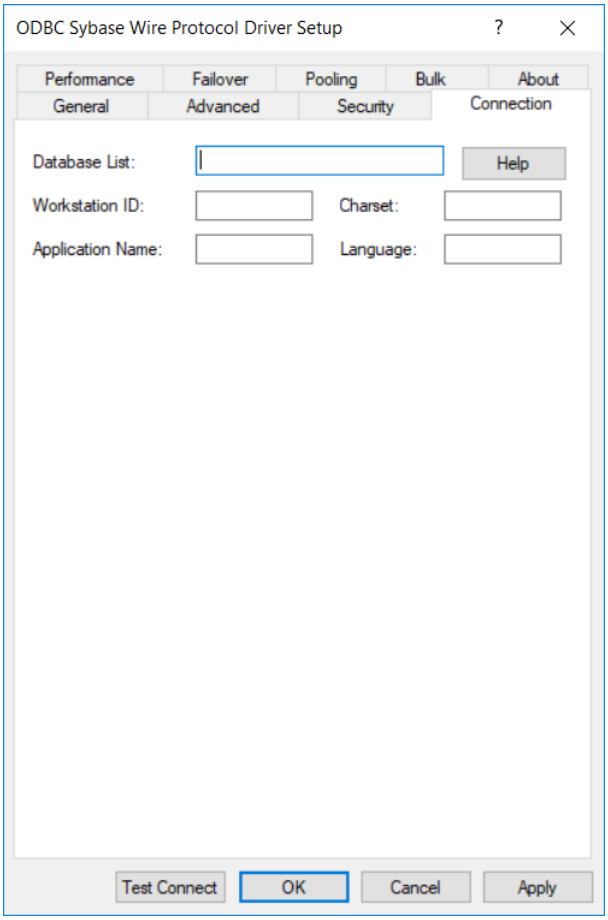
On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Security	Default
User Name on page 202	None
Authentication Method on page 159	0 - No Encryption
Service Principal Name on page 197	None
GSS Client Library on page 178	native
Encryption Method on page 173	0 - No Encryption
Crypto Protocol Version on page 165	TLSv1.2, TLSv1.1, TLSv1TLSv1, SSLv3
Validate Server Certificate on page 202	Enabled
Enable FIPS on page 119	Disabled
Truststore on page 200	None

Connection Options: Security	Default
Truststore Password on page 201	None
Host Name In Certificate on page 180	None

6. Optionally, click the **Connection** tab to specify data source settings.

Figure 9: Connection tab

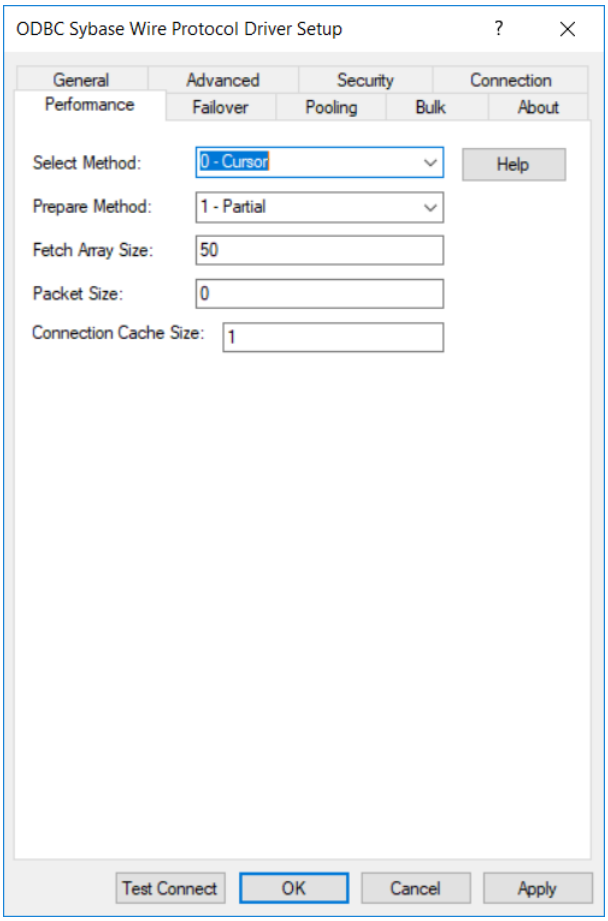


On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Connection	Default
Database List on page 168	None
Workstation ID on page 203	None
Charset on page 161	None
Application Name on page 157	None
Language on page 183	None

7. Optionally, click the **Performance** tab to specify performance data source settings.

Figure 10: Performance tab

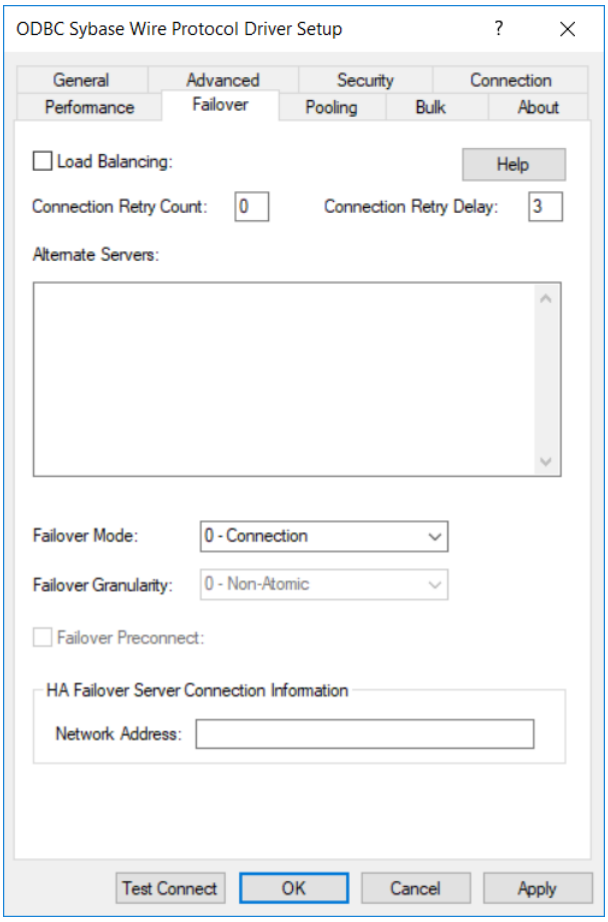


On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Performance	Default
Select Method on page 196	0 - Cursor
Prepare Method on page 190	1 - Partial
Fetch Array Size on page 176	50
Packet Size on page 189	0
Connection Cache Size on page 162	1

8. Optionally, click the **Failover** tab to specify failover data source settings.

Figure 11: Failover tab



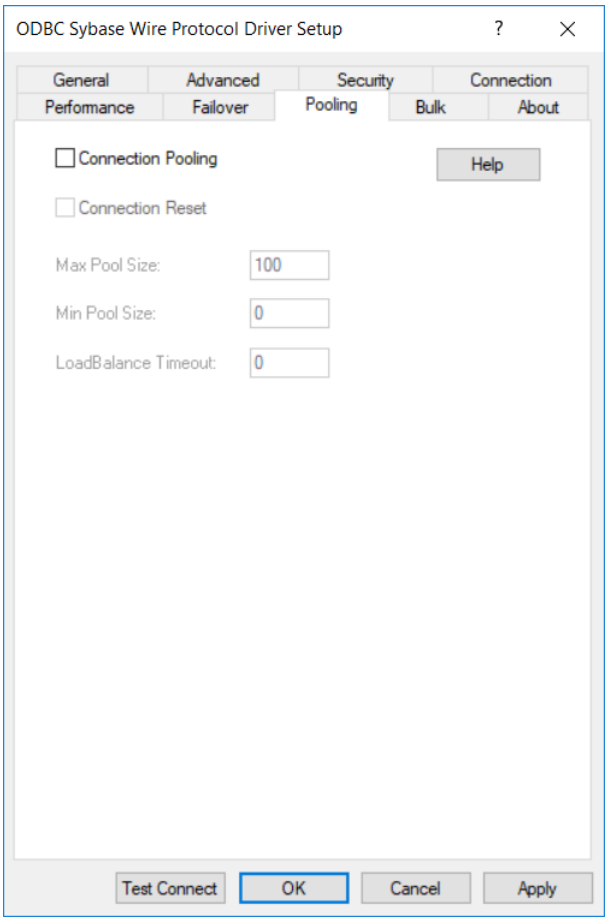
See Using Failover on page 55 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing on page 184	Disabled
Connection Retry Count on page 164	0
Connection Retry Delay on page 165	3
Alternate Servers on page 157	None
Failover Mode on page 175	0 - Connection
Failover Granularity on page 174	0 - Non-Atomic
Failover Preconnect on page 176	Disabled
HA Failover Server Connection Information/Network Address on page 179	None

9. Optionally, click the **Pooling** tab to specify connection pooling data source settings.

Figure 12: Pooling tab



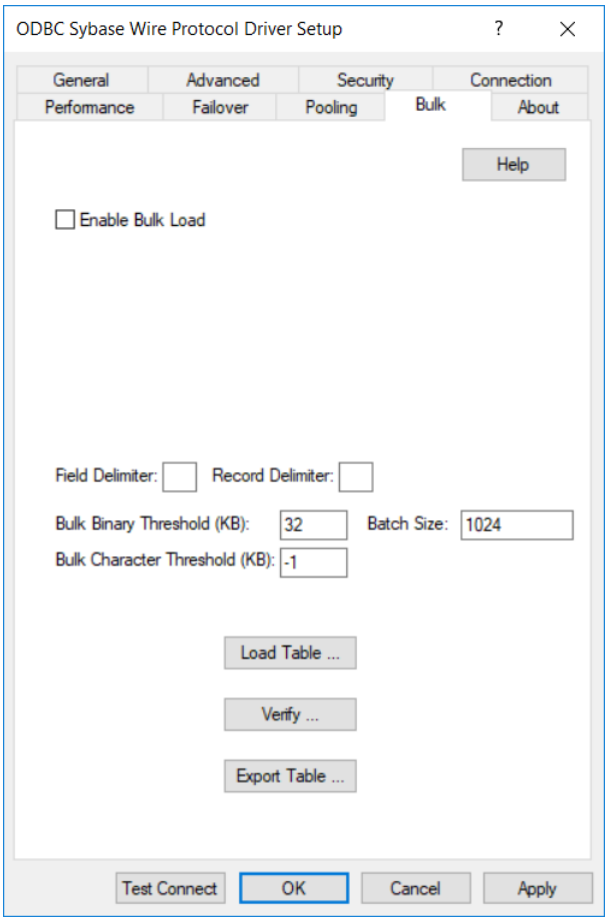
See Using DataDirect Connection Pooling on page 73 for a general description of connection pooling.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Pooling	Default
Connection Pooling on page 163	Disabled
Connection Reset on page 163	Disabled
Max Pool Size on page 186	100
Min Pool Size on page 186	0
Load Balance Timeout on page 184	0

10. Optionally, click the **Bulk** tab to specify DataDirect Bulk Load data source settings.

Figure 13: Bulk tab



See Using DataDirect Bulk Load on page 76 for a general description of DataDirect Bulk Load.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Bulk	Default
Enable Bulk Load on page 171	Disabled
Field Delimiter on page 178	None
Record Delimiter on page 194	None
Bulk Binary Threshold on page 160	32
Bulk Character Threshold on page 161	-1
Batch Size on page 159	1024

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

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

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

Figure 14: ODBC Sybase Wire Protocol Export Table Driver Setup dialog box

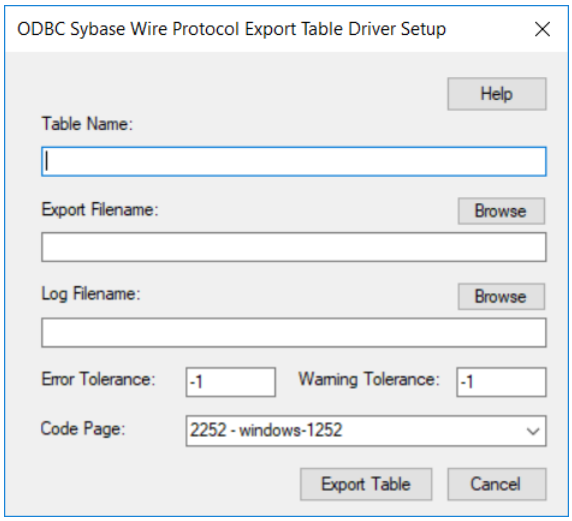


Table Name: A string that specifies the name of the source database table containing the data to be exported.

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

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. Events logged to this file are:

- Total number of rows fetched
- A message for each row that failed to export
- Total number of rows that failed to export
- Total number of rows successfully exported

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

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

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

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

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

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

Code Page: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See Character Set Conversions on page 83 for more information.

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

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

b) To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify** from the Bulk tab. See Verification of the Bulk Load Configuration File on page 82 for details. The Verify dialog box appears.

Figure 15: ODBC Sybase Wire Protocol Verify Driver Setup dialog box

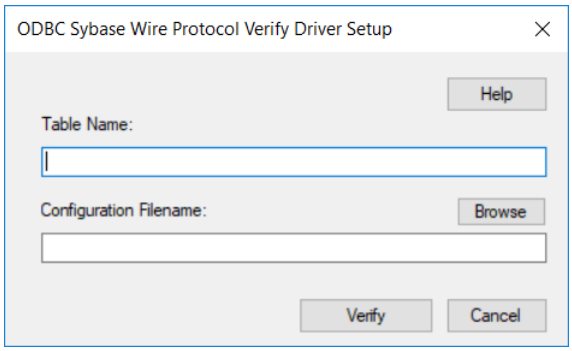


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

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

c) To bulk load data from the bulk data file to a database table, click **Load Table** from the Bulk tab. The Load File dialog box appears.

Figure 16: ODBC Sybase Wire Protocol Load File Driver Setup dialog box

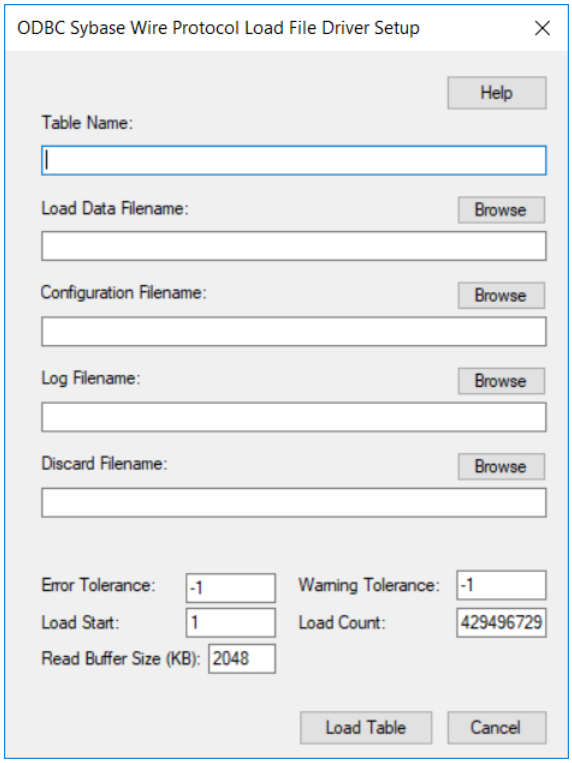


Table Name: A string that specifies the name of the target database table into which the data is loaded.

Load Data Filename: A string that specifies the path (relative or absolute) and file name of the bulk data file from which the data is loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file..

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. Specifying a value for Log Filename creates the file if it does not already exist. Events logged to this file are:

- Total number of rows read
- Message for each row that failed to load
- Total number of rows that failed to load
- Total number of rows successfully loaded

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

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

Discard Filename: A string that specifies the path (relative or absolute) and file name of the bulk discard file. The file name must be the fully qualified path to the discard file. Any row that cannot be inserted into database as result of bulk load is added to this file, with the last row rejected added to the end of the file.

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

If you do not specify a value for Discard Filename, a discard file is not created.

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

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

Load Start: A value that specifies the first row to be loaded from the data file. Rows are numbered starting with 1. For example, when Load Start is 10, the first 9 rows of the file are skipped and the first row loaded is row 10. This option can be used to restart a load after a failure.

The default value is 1.

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

The default value is 2048.

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

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

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

The default value is the maximum value for SQLULEN. If set to 0, no rows are loaded.

Click **Load Table** to connect to the database and load the table or click **Cancel**.

11. 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 (Sybase) on page 153 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

    - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
    - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

    Note: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

12. Click **OK** or **Cancel**. If 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.

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 for Sybase Wire Protocol on page 153 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 Sybase is:

```
DSN=SYB TABLES;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

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

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

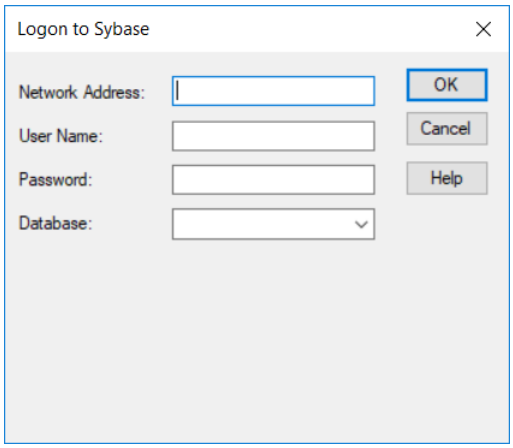
A DSN-less connection string must provide all necessary connection information:

```
DRIVER={DataDirect 7.1 Sybase Wire Protocol};NA=123.456.78.90,5000;DB=SYBACCT;
UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box (Sybase)

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

Figure 17: Logon to Sybase Dialog Box



In the Logon dialog box, provide the following information:

1. In the Network Address field, specify an IP address for the Sybase server as follows: *IP address*,*port_number*. For example, you might enter `199.226.224.34,5000`. If your network supports named servers, you can specify an address as: *servername*,*port_number*. For example, you might enter `Sybaseserver,5000`.

   The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See Using IP Addresses on page 43 for details concerning these formats.

2. If required, type your case-sensitive login ID.
3. If required, type your case-sensitive password for the system.
4. In the Database field, type the name of the database you want to access (case-sensitive). Or, select the name from the Database drop-down list, which displays the names that you specified on the Connection tab of the ODBC Sybase Wire Protocol driver Setup dialog box.

   Note: If you are connecting through the **Test Connect** button of the Setup dialog box, only the default database specified on the General tab of the Setup dialog box is available in the Database drop-down list. The database names specified on the Connection tab are not available.

5. Click **OK** to complete the logon and to update the values in the Registry.

Connection Option Descriptions for Sybase Wire Protocol

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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the Sybase Wire Protocol driver.

Table 16: Sybase Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASRV)	None
ApplicationName (APP)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	50
AuthenticationMethod (AM)	0 (No Encryption)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadBatchSize (BLBS)	1024
BulkLoadFieldDelimiter (BLFD)	None
BulkLoadRecordDelimiter (BLRD)	None
Charset (CS)	None
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CursorCacheSize (CCS)	1
CryptoLibName (CLN)	Empty string
CryptoProtocolVersion (CPV)	TLSv1.2, TLSv1.1, TLSv1, SSLv3
Database (DB)	None
Database List	None
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
Description (n/a)	None
DistributedTransactionModel (DTM)	0 (XA Protocol)

Attribute (Short Name)	Default
EnableBulkLoad (EBL)	0 (Disabled)
EnableDescribeParam (EDP)	0 (Disabled)
EnableFIPS (EF)	Disabled
EnableQuotedIdentifiers (EQI)	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverNetworkAddress (FNA)	None
FailoverPreconnect (FP)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	1
GSSClient (GSSC)	native
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
InterfacesFile (IF)	None
InterfacesFileServerName (IFSN)	None
KeepAlive (KA)	Disabled
Language	None
LoadBalanceTimeout (LBT)	None
LoadBalancing (LB)	0
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
NetworkAddress (NA)	None

Attribute (Short Name)	Default
OpenSSLConfigFile (OSSLCNF)	*install_dir*\drivers\openssl.cnf (Windows) *install_dir*/lib/openssl.cnf (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*install_dir*\drivers (Windows) *install_dir*/lib (UNIX/Linux)
OptimizePrepare (OP)	1 (Partial)
PacketSize (PS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
PRNGSeedFile (PSF) UNIX\Linux only	/dev/random
PRNGSeedSource (PSS) UNIX\Linux only	0 (File)
QueryTimeout (QT)	0
RaiserrorPositionBehavior (REPB)	0 (Default)
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
SelectMethod (SM)	0 (Cursor)
ServicePrincipalName (SPN)	None
SSLLibName (SLN)	Empty string
TightlyCoupledDistributedTransactions (TCDT)	1 (Enabled)
TruncateTimeTypeFractions (TTTF)	0 (Disabled)
Truststore (TS)	None
TruststorePassword (TSP)	Password
ValidateServerCertificate (VSC)	1 (Enabled)
WorkstationID (WKID)	None
XAOpenStringParameters (XAOSP)	None

Alternate Servers

Attribute

AlternateServers (ASRV)

Purpose

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

Valid Values

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

NetworkAddress and InterfacesFileServerName can be used in the same string.

You must specify the network address of each alternate database server or the section in the Interfaces file that contains the network connection information for the Sybase database server you want to access (InterfacesFileServerName).

Notes

- An alternate server address in IPv6 format must be enclosed in double quotation marks.
- The Alternate Servers option and the HA Failover Server Connection Information option are mutually exclusive.

Example

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

```
(InterfacesFileServerName=Accounting, NetworkAddress="255.125.1.11, 4200",
NetworkAddress="SybaseASE2, 4200")
```

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

Default

None

GUI Tab

Failover tab

Application Name

Attribute

ApplicationName (APP)

Purpose

The name used by Sybase to identify your application.

Valid Values

string

where:

`string`

is a valid application name.

Default

None

GUI Tab

Connection tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

1 (Enabled)

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 204 for details.

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

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

Valid Values

0 | 1 | 4

Behavior

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

If set to 1 (Encrypt Password), the driver negotiates a secure login with the database server by sending an encrypted password. The password is encrypted using Sybase Extended Password Encryption, Sybase Extended Plus Encrypted Password, or Sybase proprietary encryption, depending on the response by the server. Note that the proprietary Sybase password must contain characters from the 7-bit ASCII set.

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

If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Notes

- Sybase Extended Password Encryption and Sybase Extended Plus Encrypted Password, which use an asymmetrical key type, provide stronger password encryption for the secure transmission of public key passwords over networks than the proprietary encryption.

Default

0 (No Encryption)

GUI Tab

Security tab

Batch Size

Attribute

BulkLoadBatchSize (BLBS)

Purpose

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

Valid Values

0 | x

where

x

is the number of rows to send during a bulk operation.

Notes

- This connection option can affect performance.

Default

1024

GUI Tab

Bulk tab

See Also

See Performance Considerations on page 204 for details.

Bulk Binary Threshold

Attribute

BulkBinaryThreshold (BBT)

Purpose

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

Valid Values

-1 | 0 | *x*

where

x

is an integer that specifies the number of KB.

Behavior

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

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

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

Notes

- This connection option can affect performance.

Default

32

GUI Tab

Bulk tab

See Also

See Performance Considerations on page 204 for details.

Bulk Character Threshold

Attribute

BulkCharacterThreshold (BCT)

Purpose

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

Valid Values

-1 | 0 | *x*

where:

x

is an integer that specifies the number of KB.

Behavior

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

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

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

Default

-1

GUI Tab

Bulk tab

Charset

Attribute

Charset (CS)

Purpose

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

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

Valid Values

charset

where:

charset

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

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

For the driver to return Unicode SQL types for connections to Sybase 12.5 and higher, use a value of UTF-8. Refer to the Sybase server documentation for a list of valid character sets.

Example

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

Default

None

GUI Tab

Connection tab

Connection Cache Size

Attribute

CursorCacheSize (CCS)

Purpose

The number of connections that the connection cache can hold.

Valid Values

x

where:

x

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

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

Default

1

GUI Tab

Performance tab

Connection Pooling

Attribute

Pooling (POOL)

Purpose

Specifies whether to use the driver's connection pooling.

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

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Pooling tab

See Also

See Performance Considerations on page 204 for details.

Connection Reset

Attribute

ConnectionReset (CR)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Pooling tab

See Also

See Performance Considerations on page 204 for details.

Connection Retry Count

Attribute

ConnectionRetryCount (CRC)

Purpose

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

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

Valid Values

0 | *x*

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

0

GUI Tab

Failover tab

Connection Retry Delay

Attribute

ConnectionRetryDelay (CRD)

Purpose

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

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

Valid Values

0 | *x*

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

3

GUI Tab

Failover tab

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.2 | TLSv1.3

Example

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

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

Notes

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

Default

TLSv1.2,TLSv1.1

GUI Tab

Security tab

See also

Encryption Method on page 173

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

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

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

Valid Values

absolute_path\openssl_filename

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

Empty string

GUI Tab

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

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

See Advanced tab for details.

See also

SSLLibName on page 197

Cursor Positioning for Raiserror

Attribute

RaiserrorPositionBehavior (REPB)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

0 (Default)

GUI Tab

Advanced tab

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

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

Valid Values

string

where:

string

is the name of a data source.

Default

None

GUI Tab

General tab

Database List

Attribute

n/a

Purpose

A list of database names that will appear in the drop-down list of the logon dialog box (see Using a Logon Dialog Box (Sybase) on page 153 for a description).

Valid Values

database_list

where:

database_list

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

Default

None

GUI Tab

Connection tab

Database Name

Attribute

Database (DB)

Purpose

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

Valid Values

database_name

where:

database_name

is the name of a valid database.

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

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute

DefaultLongDataBuffLen (DLDBL)

Purpose

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

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

Valid Values

An integer in multiples of 1024

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

Notes

- This connection option can affect performance.

Default

1024

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 204 for details.

Distributed Transaction Model

Attribute

DistributedTransactionModel (DTM)

Purpose

The model to use for distributed transaction support. The driver supports two different models: XA Protocol and Native OLE.

Valid Values

0 | 1

Specify the appropriate distributed transaction protocol, either 0 (XA Protocol) or 1 (Native OLE)

Default

0 (XA Protocol)

GUI Tab

Advanced tab

Enable Bulk Load

Attribute

EnableBulkLoad (EBL)

Purpose

Specifies the bulk load method.

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Bulk tab

See Also

See Performance Considerations on page 204 for details.

Enable Describe Parameter

Attribute

EnableDescribeParam (EDP)

Purpose

Determines whether the driver supports the SQLDescribeParam function, which allows an application to describe parameters in SQL statements and in stored procedure calls.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver supports SQLDescribeParam. The Prepare Method option must be set to 0 or 1, and the SQL statement must not include long parameters. If using Microsoft Remote Data Objects (RDO) to access data, you must use this value.

If set to 0 (Disabled), the driver does not support SQLDescribeParam.

Default

0 (Disabled)

GUI Tab

Advanced tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

0

Enable Quoted Identifiers

Attribute

EnableQuotedIdentifiers (EQI)

Purpose

Determines whether the driver supports the use of quoted identifiers.

Valid Values

0 | 1

If set to 1 (Enabled), the driver supports the use of quoted identifiers. Double quotation marks (") must be used to enclose identifiers, such as column and table names. Character strings must be enclosed in single quotation marks, for example:

```
SELECT "au_id"
FROM "authors"
WHERE "au_lname" = 'O''Brien'
```

If set to 0 (Disabled), the driver does not support the use of quoted identifiers and generates an error when quoted identifiers are encountered.

Default

0 (Disabled)

GUI Tab

Advanced tab

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

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

Valid Values

0 | 1

Behavior

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

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.

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

Notes

- This connection option can affect performance.

Default

0 (No Encryption)

GUI Tab

Security tab

See Also

Crypto Protocol Version on page 165

Performance Considerations on page 204

Failover Granularity

Attribute

FailoverGranularity (FG)

Purpose

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

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

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

Valid Values

0 | 1 | 2 | 3

Behavior

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

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

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

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

Default

0 (Non-Atomic)

GUI Tab

Failover tab

Failover Mode

Attribute

FailoverMode (FM)

Purpose

Specifies the type of failover method the driver uses.

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

Notes

- This connection option can affect performance.

Default

0 (Connection)

GUI Tab

Failover tab

See Also

See Performance Considerations on page 204 for details.

Failover Preconnect

Attribute

FailoverPreconnect (FP)

Purpose

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

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

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

Valid Values

0 | 1

Behavior

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

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

Default

0 (Disabled)

GUI Tab

Failover tab

Fetch Array Size

Attribute

ArraySize (AS)

Purpose

The number of rows the driver retrieves from the server for a fetch. This is not the number of rows given to the user. This connection option can affect performance.

Valid Values

x

where:

x

is a positive integer specifying the number of rows.

Notes

- This connection option can affect performance.

Default

50

GUI Tab

Performance tab

See Also

See Performance Considerations on page 204 for details.

Fetch TWFS as Time

Attribute

FetchTWFSasTime (FTWFSAT)

Purpose

Determines which ODBC data type the driver uses to return column values with the BIGTIME data type.

Valid Values

0 | 1

Behavior

If set to 1, the driver returns column values for the BIGTIME data type as the ODBC data type SQL_TYPE_TIME. The fractional seconds portion of the value is truncated.

If set to 0, the driver returns column values for the BIGTIME data type as the ODBC data type SQL_TYPE_TIMESTAMP. When a timestamp is returned for BIGTIME, the Year, Month and Day parts of the timestamp must be set to zero.

Notes

- The BIGTIME data type is supported in Sybase 15.5 and higher.

Default

1

GUI Tab

Advanced tab

Field Delimiter

Attribute

BulkLoadFieldDelimiter (BLFD)

Purpose

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

Valid Values

x

where:

x

is any printable character.

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

Notes

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

Default

None

GUI Tab

Bulk tab

GSS Client Library

Attribute

GSSClient (GSSC)

Purpose

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

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

Valid Values

native | *`client_library`*

where:

client_library

is a GSS client library installed on the client.

Behavior

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

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

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the GSS Client Library option. The GSS Client Library option provides a method for you to specify a library file used to communicate with the Key Distribution Center (KDC). However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.

Default

native

GUI Tab

Security tab

HA Failover Server Connection Information/Network Address

Attribute

FailoverNetworkAddress (FNA)

Purpose

The network address of the High Availability (HA) Failover server to be used in the event of a connection loss. The driver detects the dropped connection and automatically reconnects to the specified HA Failover server. This option is valid only for Sybase 12 and higher servers that have the High Availability Failover feature enabled.

Valid Values

IP_*address*, *port_number* | *pipe_address*, *port_number* | *server_name*, *port_number*

where:

IP_ address

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

port_number

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

server_name

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

pipe_address

is the pipe address of the HA Failover server. This format is required if using NamedPipes as the network protocol.

Notes

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

Example

```
199.226.224.34, 5000
```

or

```
\\machine1\sybase\pipe\query, 5000
```

or

```
Sybaseserver, 5000
```

Default

None

GUI Tab

Failover tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

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

Valid Values

host_name | *#SERVERNAME#*

where:

host_name

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

Behavior

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

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

Default

None

GUI Tab

Security tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

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

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

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

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

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

Valid Values

IANA_code_page

where:

IANA_code_page

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

Notes

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

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

Advanced tab

Initialization String

Attribute

InitializationString (IS)

Purpose

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

Valid Values

SQL_command

where:

SQL_command

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

Notes

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

Example

To allow delimited identifiers, specify:

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

Default

None

GUI Tab

Advanced tab

Interfaces File

Attribute

InterfacesFile (IF)

Purpose

The directory to the Interfaces file.

Valid Values

file_dir

where:

file_dir

is the directory to the Interfaces file.

Behavior

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

Notes

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

Default

None

GUI Tab

General tab

Language

Attribute

Language (LANG)

Purpose

The national character set installed on the Sybase server.

Valid Values

charset

where:

charset

is the national character set installed on the Sybase server.

Default

None (English)

GUI Tab

Connection tab

Load Balance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

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

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

Valid Values

0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to 0, inactive connections are kept open.

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

Notes

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

Default

0 (Disabled)

See also

See Performance Considerations on page 204 for details.

GUI Tab

Pooling tab

Load Balancing

Attribute

LoadBalancing (LB)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

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

Default

0 (Disabled)

GUI Tab

Failover tab

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

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

Valid Values

-1 | 0 | x

where:

x

is a positive integer that represents a number of seconds.

Behavior

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

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

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

Default

15

GUI Tab

Advanced tab

Max Pool Size

Attribute

MaxPoolSize (MXPS)

Purpose

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

Valid Values

An integer from 1 to 65535

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

Notes

- This connection option can affect performance.

Default

100

GUI Tab

Pooling tab

See Also

See Performance Considerations on page 204 for details.

Min Pool Size

Attribute

MinPoolSize (MNPS)

Purpose

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

This connection option can affect performance. See Performance Considerations for details.

Valid Values

0 | *x*

Behavior

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

Notes

- This connection option can affect performance.

Example

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

Default

0

GUI Tab

Pooling tab

See Also

See Performance Considerations on page 204 for details.

Network Address

Attribute

NetworkAddress (NA)

Purpose

A unique identifier assigned to the Sybase server machine.

Valid Values

server_name | *IP_address*

where:

server_name

is the Sybase server name specified as: *named_server*, *port_number*. For example, you can enter `SSserver, 5000`.

IP_address

is the Sybase server address specified as: *IP_address*, *port_number*. For example, you can enter `199.226.224.34, 5000`. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See Using IP Addresses on page 43 for details about these formats.

Notes

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

Default

None

GUI Tab

General tab

OpenSSLConfigFile

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

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

Valid Values

fips_config_file

where:

fips_config_file

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

Notes

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

Default

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

OpenSSLProviderPath

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

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

Valid Values

provider_path

where:

provider_path

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

Notes

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

Default

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

Packet Size

Attribute

PacketSize (PS)

Purpose

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

Valid Values

-1 | 0 | *x*

Behavior

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

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

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

Notes

- If SSL encryption is used, the driver must use the packet size that is specified by the server. Any value set for this option or the SQL_PACKET_SIZE connect option is ignored if SSL encryption is used.

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

Default

0

GUI Tab

Performance tab

See Also

See Performance Considerations on page 204 for details.

Password

Attribute

Password (PWD)

Purpose

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

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Prepare Method

Attribute

OptimizePrepare (OP)

Purpose

Determines whether stored procedures are created on the server for calls to SQLPrepare.

Valid Values

0 | 1 | 2 | 3

Behavior

If set to 0 - None, stored procedures are created for every call to SQLPrepare. This setting can result in decreased performance when processing statements that do not contain parameters.

If set to 1 - Partial, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and run directly at the time of SQLExecute.

If set to 2 - Full, stored procedures are never created. The driver caches the statement, executes it directly at the time of SQLExecute, and reports any syntax or similar errors at the time of SQLExecute.

If set to 3 - Full at Prepare, stored procedures are never created. This is identical to value 2 except that any syntax or similar errors are returned at the time of SQLPrepare instead of SQLExecute. Use this setting only if you must have syntax errors reported at the time of SQLPrepare.

Notes

- This connection option can affect performance.

Default

1 (Partial)

GUI Tab

Performance tab

See Also

See Performance Considerations on page 204 for details.

PRNGSeedFile

Attribute

PRNGSeedFile (PSF)

Purpose

UNIX® Specifies the absolute path for the entropy-source file or device used as a seed for TLS/SSL key generation.

Valid Values

`string` | `RANDFILE`

where:

`string`

is the absolute path for the entropy-source file or device that seeds the random number generator used for TLS/SSL key generation.

Behavior

If set to *`string`*, the specified entropy-source file or device seeds the random number generator used for TLS/SSL key generation. Entropy levels and behavior may vary for different files and devices. See the following section for a list of commonly used entropy sources and their behavior.

If set to `RANDFILE`, the `RAND_file_name()` function in your application generates a default path for the random seed file. The seed file is `$RANDFILE` if that environment variable is set; otherwise, it is `$HOME/.rnd`. If `$HOME` is not set either, an error occurs.

Common Valid Values

Although other entropy-source files may be specified, the following valid values are for files and devices that are commonly used for seeding:

`/dev/random`

is a pseudorandom number generator (blocking) that creates a seed from random bits of environmental noise it collects in an entropy pool. When there is insufficient noise in the pool, the file blocks calls until enough noise is collected. This provides more secure TLS/SSL key generation, but at the expense of blocked calls.

`/dev/urandom`

is a pseudorandom number generator (non-blocking) that creates seeds from random bits from environmental noise it collects in an entropy pool. When there is insufficient noise in the pool, the file reuses bits from the pool instead of blocking calls. This eliminates potential delays associated with blocked calls, but may result in less secure TLS/SSL key generation.

`/dev/hwrng`

is a hardware random number generator. The behavior is dependent on the device used in your environment.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the PRNGSeedFile option. The PRNGSeedFile option provides a method for you to specify a entropy-source file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- This option is ignored when TLS/SSL is disabled (`EncryptionMethod=0`) or the seed source is set to Poll Only (`PRNGSeedSource=1`).
- For processes that employ multiple TLS/SSL-enabled drivers, the behavior of this option for all drivers is determined by the values specified for the driver that first connects to the process and loads the OpenSSL library. Since the OpenSSL library loads only once per process, the values specified for drivers that subsequently connect are ignored. To ensure that the correct security settings are used, we recommend configuring this option identically for all drivers used in a process.

Default

`/dev/random`

GUI tab

NA

See also

PRNGSeedSource on page 193

PRNGSeedSource

Attribute

PRNGSeedSource (PSS)

Purpose

UNIX® Specifies the source of the seed the driver uses for TLS/SSL key generation. Seeds are a pseudorandom or random value used to set the initial state of the random number generator used to generate TLS/SSL keys. Using seeds with a higher level of entropy, or randomness, provides a more secure transmission of data encrypted using TLS/SSL.

Valid Values

`0` | `1`

Behavior

If set to `0` (File), the driver uses entropy-source file or device specified in the PRNGSeedFile connection option as the seed used for TLS/SSL key generation.

If set to `1` (Poll Only) , the driver uses the RAND_poll function in TLS/SSL to create the seed used for TLS/SSL key generation.

Notes

- For processes that employ multiple TLS/SSL-enabled drivers, the behavior of this option for all drivers is determined by the values specified for the driver that first connects to the process and loads the OpenSSL library. Since the OpenSSL library loads only once per process, the values specified for drivers that subsequently connect are ignored. To ensure that the correct security settings are used, we recommend configuring this option identically for all drivers used in a process.
- This option is ignored when TLS/SSL is disabled (`EncryptionMethod=0`)

Default

`0` (File)

GUI Tab

NA

See also

PRNGSeedFile on page 191

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

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

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

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

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

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

Default

0

GUI Tab

Advanced tab

Record Delimiter

Attribute

BulkLoadRecordDelimiter (BLRD)

Purpose

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

Valid Values

x

where:

x

is any printable character.

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

Notes

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

Default

None

GUI Tab

Bulk tab

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

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

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

Default

0 (Ignore Errors)

GUI Tab

Advanced tab

Select Method

Attribute

SelectMethod (SM)

Purpose

Determines whether database cursors are used for Select statements.

Valid Values

0 | 1

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

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

Notes

- This connection option can affect performance.

Default

0 (Cursor)

GUI Tab

Performance tab

See Also

See Performance Considerations on page 204 for details.

Server Name

Attribute

InterfacesFileServerName (IFSN)

Purpose

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

Valid Values

section_name

where:

section_name

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

Notes

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

Default

None

GUI Tab

General tab

Service Principal Name

Attribute

ServicePrincipalName (SPN)

Purpose

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

Valid Values

servicePrincipalName

where:

servicePrincipalName

is a valid service principal name.

If unspecified, the value of the Network Address option is used as the service principal name. If Authentication Method is set to 0 or 1, the value of the Service Principal Name option is ignored.

Default

None

GUI Tab

Security tab

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

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

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

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

No default value

GUI Tab

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

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

See Advanced tab for details.

See also

CryptoLibName on page 166

TCP Keep Alive

Attribute

KeepAlive (KA)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

0 (Disabled)

GUI Tab

Advanced tab

Tightly Coupled Distributed Transactions

Attribute

TightlyCoupledDistributedTransactions (TCDT)

Purpose

Sybase 12 or higher server only. Determines whether the driver ensures that multiple connections within the same distributed transaction obey other's locks.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver uses tightly coupled distributed transactions. Multiple connections within the same distributed transaction obey other's locks.

If set to 0 (Disabled), the driver does not use tightly coupled distributed transactions. Multiple connections within the same distributed transaction may hang each other because the connections do not obey other's locks. This value can provide better performance if concurrency of data is not needed.

Default

1 (Enabled)

GUI Tab

Advanced tab

Truncate Time Type Fractions

Attribute

TruncateTimeTypeFractions (TTTF)

Purpose

Sybase 12.5.1 and higher only. Determines whether the driver sets fractional seconds to zero (0) when converting data from the TIME data type to TIMESTAMP, CHAR, or WCHAR data types.

Valid Values

0 | 1

Behavior

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

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

Default

0 (Disabled)

GUI Tab

Advanced tab

Truststore

Attribute

Truststore (TS)

Purpose

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

Valid Values

*truststore_directory**filename*

where:

truststore_directory

is the directory where the truststore file is located

filename

is the file name of the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

Truststore Password

Attribute

TruststorePassword (TSP)

Purpose

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

Valid Values

truststore_password

where:

truststore_password

is a valid password for the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

User Name

Attribute

LogonID (UID)

Purpose

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

Valid Values

userid

where:

userid

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

Default

None

GUI Tab

Security tab

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

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

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI Tab

Security tab

Workstation ID

Attribute

WorkstationID (WKID)

Purpose

An identifier for the client machine.

Valid Values

ID

where:

ID

is workstation ID use by the client machine.

Default

None

GUI Tab

Connection tab

XA Open String Parameters

Attribute

XAOpenStringParameters (XAOSP)

Purpose

Determines the name of trace files generated for XA open string parameters.

Valid Values

`-Ltrace_`*`filename`*

where:

`trace_ filename`

is a string that identifies trace files generated for XA open string parameters. If specified, two trace files are created. The first trace file traces all XA call activities and is named exactly as specified. The second trace file traces any enlistment and unenlistment procedures and is named as specified with a "driver" extension.

Example

If you specify `-LXAtrace`, the driver creates two trace files: XAtrace and XAtrace.driver.

Default

None

GUI Tab

Advanced tab

Performance Considerations

The following connection options can enhance driver performance.

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

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

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

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

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

Enable Bulk Load (**EnableBulkLoad**): If your application performs bulk loading of data, you can improve performance by configuring the driver to use the database system's bulk load functionality instead of database array binding. The trade-off to consider for improved performance is that using the bulk load functionality can bypass data integrity constraints.

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

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

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

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

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

Prepare Method (OptimizePrepare): If your application executes the same SQL statements multiple times, performance can be improved by creating a stored procedure on the server at prepare time. If your application executes one of these prepared statements multiple times, performance will improve because the driver created a stored procedure and executing a stored procedure is faster than executing a single SQL statement; however, if a prepared statement is only executed once or is never executed, performance can decrease. If your application executes the same SQL statements multiple times, the Prepare Method option should be set to 1.

Select Method (SelectMethod): If your application often executes a SQL statement before processing or closing the previous result set, then it uses multiple active statements per connection. The default setting (0) of this option causes the driver to use database cursors for Select statements and allows an application to process multiple active statements per connection. An active statement is defined as a statement where all the result rows or result sets have not been fetched. This can cause high overhead on the server. If your application does not use multiple active statements, however, setting Select Method to 1 will increase performance of Select statements by allowing the server to return results without using a database cursor. If this option is set to 0, it should be used in conjunction with Fetch Array Size (ArraySize). If this option is set to 1, Fetch Array Size (ArraySize) has no effect.

Data Types

The following table shows how the Sybase data types are mapped to the standard ODBC data types. Unicode Support on page 207 lists Sybase to Unicode data type mappings.

Table 17: Sybase Data Type Mapping

Sybase Data Type...	Maps to ODBC Data Type
BIGDATETIME[1]	SQL_DATETIME
BIGINT [2]	SQL_BIGINT
BIGTIME[1]	SQL_DATETIME
BINARY	SQL_BINARY
BIT	SQL_BIT
CHAR	SQL_CHAR
DATE[3]	SQL_TYPE_DATE
DATETIME	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
FLOAT	SQL_FLOAT
IMAGE	SQL_LONGVARBINARY
INT	SQL_INTEGER
MONEY	SQL_DECIMAL
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLDATETIME	SQL_TYPE_TIMESTAMP
SMALLINT	SQL_SMALLINT
SMALLMONEY	SQL_DECIMAL
SYSNAME	SQL_VARCHAR
TEXT	SQL_LONGVARCHAR
TIME[3]	SQL_TYPE_TIME
TIMESTAMP	SQL_VARBINARY
TINYINT	SQL_TINYINT

[1] Sybase 15.5 and higher only.
[2] Sybase 15 and higher only.
[3] Sybase 12.5.1 and higher only.

Sybase Data Type...	Maps to ODBC Data Type
UNSIGNED BIGINT2	SQL_BIGINT
UNSIGNED INT2	SQL_INTEGER
UNSIGNED SMALLINT[2]	SQL_SMALLINT
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_VARCHAR

Note: FOR USERS OF SYBASE 12.5 and higher: The Sybase Wire Protocol driver supports extended new limits (XNL) for character and binary columns—columns with lengths greater than 255.

See Retrieving Data Type Information on page 47 for information about retrieving data types.

Unicode Support

When connected to a Unicode database, the Sybase Wire Protocol driver supports Unicode data types listed in the following table, in addition to standard ODBC data types listed in Data Types on page 205.

Sybase Data Type	Mapped to. . .
CHAR[4]	SQL_WCHAR
SYSNAME[4]	SQL_VARCHAR
TEXT[4]	SQL_WLONGVARCHAR
UNICHAR[5]	SQL_WCHAR
UNITEXT[6]	SQL_WLONGVARCHAR
UNIVARCHAR[5]	SQL_WVARCHAR
VARCHAR [4]	SQL_WVARCHAR

For data types that require the UTF-8 character set, set the Charset connection string attribute. See Charset on page 161 for information about using this connection string attribute.

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

See UTF-16 Applications on UNIX and Linux on page 101 for related details.

[4] This data type is available only if the data source is configured to use the UTF-8 character set.
[5] On Sybase 12.5 servers, this data type is available only if the data source is configured to use the UTF-8 character set. On Sybase 12.5.1 and higher servers, this data type is always available, even if the data source is not configured to use the UTF-8 character set.
[6] This data type is available on Sybase 15 and higher servers only.

Also, refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for a more detailed explanation of Unicode.

Advanced Features

The driver supports the following advanced features:

- Failover
- Security
- Connection Pooling
- DataDirect Bulk Load

Failover

The driver supports failover and its related connection options. Failover connection options are located on the Failover tab of the driver Setup dialog box. See Using Failover on page 55 for a general description of failover and its implementation.

Security

The driver supports authentication and encryption. Security connection options are located on the Security tab of the driver Setup dialog box. See Using Security on page 63 for a general description of security and its implementation. The following security information is specific to the Sybase Wire Protocol Driver.

Authentication

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

Table 18: Kerberos Authentication Requirements for the Sybase Wire Protocol Driver

Component	Requirements
Database server	The database server must be administered by the same domain controller that administers the client and must be running Sybase 12.0 or higher. In addition, the Sybase Security and directory services package, ASE_SECDIR, is required.
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by one of the following methods: • Windows Active Directory • MIT Kerberos 1.4.2 or higher
Client	The client must be administered by the same domain controller that administers the database server.

Connection Pooling

The driver supports connection pooling and its related connection options. Connection pooling connection options are located on the Pooling tab of the driver Setup dialog box. See Using DataDirect Connection Pooling on page 73 for a general description of connection pooling and its implementation.

DataDirect Bulk Load

The driver supports DataDirect bulk load and its related connection options. Bulk load connection options are located on the Bulk tab of the driver Setup dialog box. See Using DataDirect Bulk Load on page 76 for a general description of DataDirect bulk load and its implementation.

For Sybase, some additional database configuration is required when the destination table for a bulk load operation does not have an index defined. If you are using a destination table that does not have an index defined, you can ask the database operator to execute the following commands:

```
use master
sp_dboption test, "select into/bulkcopy/pllsort", true
```

This option is required to perform operations that do not keep a complete record of the transaction in the log. For more information, refer to the Sybase documentation.

Alternatively, you can define an index on the destination table.

Failure to properly configure the database results in errors such as the following:

```
"You cannot run the non-logged version of bulk copy in this database. Please check with
 the DBO."
```

Bulk Copy Operations and Transactions

Sybase does not support a bulk insert within a transaction, and returns an error if a bulk copy operation is attempted in the scope of an existing transaction.

The Sybase server treats each batch of the bulk copy operation as a single transaction. If any rows in the batch are rejected, the entire transaction is rolled back.

Limitations

- The Sybase server ensures the accuracy of Real data type values only up to 6 digits.
- For the Money data type, if an application submits a value with a scale larger than 4, the driver changes the scale to 4 by truncating the value. Ideally, it should set the scale to 4 and round off the value. For example, if the value is 100.141592, the driver truncates it to 100.1415. Instead, it should round it off to 100.1416.
- The driver does not support inserting data containing LOB columns. In such cases, the driver throws a warning and falls back to the native protocol to continue executing the inserts.
- When executing an insert statement, the operation will fail with an error message if non-identity columns are omitted from a statement.

Performance Considerations

Sybase defines two bulk copy modes, described in the following table. Sybase automatically selects the appropriate mode at run time. For more information, refer to your Sybase documentation.

Table 19: Summary of Fast and Slow Bulk Copy Mode Characteristics

Characteristic	Fast Bulk Copy Mode	Slow Bulk Copy Mode
Destination Table Characteristics	No indexes or triggers on destination table	One or more indexes or triggers
Database Configuration Required	The into/bulkcopy/pllsort dboption must be set to true.	None
Logging Performed	Page allocations are logged, but row inserts are not	Row inserts are logged
Transaction Log Handling	You must dump the database before backing up (dumping) the transaction log.	The transaction log can become very large. After the bulk copy completes, back up your database with dump database, then truncate the log with dump transaction.

Unexpected Characters

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

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

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage attribute (see IANAAppCodePage on page 181). If it does not find a specific setting for IACP, it defaults to a value of ISO_8859_1.

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

The following is a specific example of an unexpected character:

- The Windows client machine is running code page cp1252.
- The Sybase server is running code page cp850.
- You insert decimal literals for character data. You think you are inserting LATIN SMALL LETTER I WITH ACUTE (í) and BOX DRAWINGS DOUBLE VERTICAL (||) in the database. When you fetch the data, you

see INVERTED EXCLAMATION MARK (¡) and MASCULINE ORDINAL INDICATOR (º) displayed on the client instead.

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

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

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

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

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 DataDirect Connect for ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 for ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

To enable DTC support, you must be accessing Sybase Adaptive Server Enterprise 12.0 or higher. You can choose either Native OLE and XA protocol distributed transactions. See the Distributed Transaction Model option documented in Configuring and Connecting to Data Sources on page 138 for details.

To enable distributed transaction in the Sybase server:

1. Assign the dtm_tm_role to each user who will participate in distributed transactions (who will log in to Adaptive Server). You can do this using the sp_role command. For example:

   ```
   sp_role "grant", dtm_tm_role, user_name
   ```

   In the open string for resource managers, the specified username must have the dtm_tm_role.

2. Specify a default database other than the master for each user. Sybase cannot start distributed transactions in a master database.

NULL Values

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

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

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

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

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

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

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

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

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

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

  ```
  SET ANSINULL OFF
  ```

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See Persisting a Result Set as an XML Data File on page 48 for details about implementation.

Isolation and Lock Levels Supported

The Sybase database system supports isolation levels 0 (read uncommitted), 1 (read committed, the default), 2 (repeatable read), and 3 (serializable). It supports page-level locking.

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

SQL Grammar Support

The driver supports the minimum SQL grammar.

ODBC Conformance Level

The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions. In addition, the following functions are supported:

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

- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

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

Number of Connections and Statements Supported

The Sybase database system supports multiple connections and multiple statements per connection. If the Select Method option on the Performance tab or the connection string attribute SelectMethod is set to 1 (Direct), Sybase data sources are limited to one active statement in manual commit mode.

Using Arrays of Parameters

When designing an application, using parameter arrays for bulk inserts or updates, for example, can improve performance.

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

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

The Text Driver

The DataDirect Connect for ODBC and DataDirect Connect64 for ODBC Text driver (the Text driver) supports ASCII text files.

These files can be printed directly or edited with text editors or word processors, because none of the data is stored in a binary format.

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

The 32-bit and 64-bit Text driver is is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

The Text driver executes SQL statements directly on the text files. The driver supports Insert statements and inserts the record at the end of the file. You can execute Update and Delete statements conditionally.

The Text driver can access files up to 15 GB in size.

Refer to the readme file shipped with your DataDirect Connect product for the file names for the Text driver.

Driver Requirements

There are no client requirements for the Text driver.

Formats for Text Files

Some common formats for text files are listed in the following table.

Table 20: Text File Formats

Format	Description
Comma-separated values	Commas separate column values, and each line is a separate record. Column values can vary in length. These files often have the .CSV extension.
Tab-separated values	Tabs separate column values, and each line is a separate record. Column values can vary in length.
Character-separated values	Any printable character except single and double quotes can separate column values, and each line is a separate record. Column values can vary in length.
Fixed	No character separates column values. Instead, values start at the same position and have the same length in each line. The values appear in fixed columns if you display the file. Each line is a separate record.
Stream	No character separates column values nor records. The table is one long stream of bytes.

Comma-, tab-, and character-separated files are called character-delimited files because values are separated by a special character.

Configuring Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 218 and Connection Option Descriptions on page 219 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX/Linux odbc.ini File

UNIX® On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration on page 27 for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Connection Option Descriptions on page 219 lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (Text)

 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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

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

To configure a Text data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

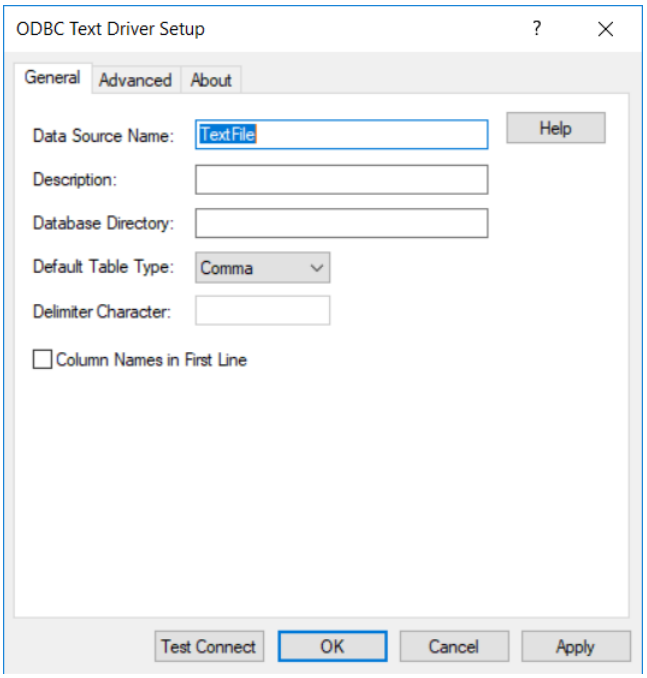
     If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

   - **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.

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

Figure 18: General tab



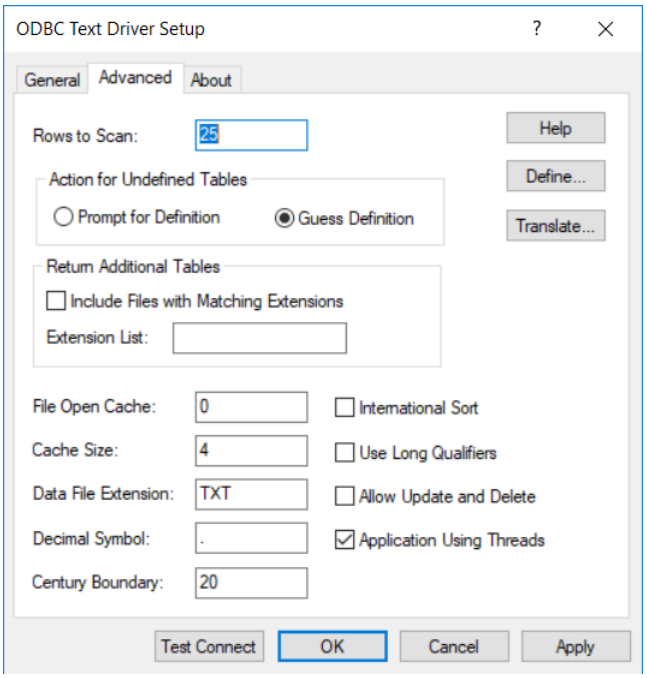
NOTE: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3. On this tab, provide values for the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 224	None
Description on page 227	None
Database Directory on page 225	None
Default Table Type on page 226	Comma
Delimiter Character on page 226	None
Column Names in First Line on page 223	Disabled

4. Optionally, click the **Advanced** tab to specify data source settings.

Figure 19: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Rows to Scan on page 230	25
Action for Undefined Tables on page 220	GUESS
Include Files with Matching Extensions on page 229	Disabled
Extension List on page 227	None
File Open Cache on page 228	0
International Sort on page 230	Disabled
Cache Size on page 222	4
Use Long Qualifiers on page 231 WINDOWS ONLY	Disabled
Data File Extension on page 223	TXT
Allow Update And Delete on page 220	Disabled
Decimal Symbol on page 225	. (Period)
Application Using Threads on page 221	Enabled

Connection Options: Advanced	Default
Century Boundary on page 222	20
IANAAppCodePage on page 229 UNIX ONLY	4 (ISO 8559-1 Latin-1)

Define: Click **Define** to define the structure of your text files as described in Defining Table Structure on Windows on page 232.

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

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

5. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.
   - If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.
6. Click **OK** or **Cancel**. If 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.

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 on page 219 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 Text is:

```
DSN=Text1;DB=C:\TEXTDATA;TT=Comma
```

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

```
FILEDSN=Text1.dsn;DB=C:\TEXTDATA;TT=Comma
```

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

```
DRIVER={DataDirect 7.1 TextFile (*.*)};DB=C:\TEXTDATA;TT=Comma
```

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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the Text driver.

Table 21: Text Attribute Names

Attribute (Short Name)	Default
AllowUpdateAndDelete (AUD)	0 (Disabled)
ApplicationUsingThreads (AUT)	1 (Enabled)
CacheSize (CSZ)	4
CenturyBoundary (CB)	20
Database (DB)	None
DataFileExtension (DFE)	TXT
DataSourceName (DSN)	None
DecimalSymbol (CS)	. (Period)
Delimiter (DC)	, (Comma)
Description (n/a)	None
ExtraExtensions (EE) WINDOWS ONLY	None
FileOpenCache (FOC)	0

Attribute (Short Name)	Default
FirstLineNames (FLN)	0 (Disabled)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntlSort (IS)	0 (Disabled)
ScanRows (SR)	25
TableType (TT)	Comma
UndefinedTable (UT))	GUESS
UseLongQualifiers (ULQ) WINDOWS ONLY	0 (Disabled)

Action for Undefined Tables

Attribute

UndefinedTable (UT)

Purpose

Determines whether the driver prompts the user when it encounters a table for which it has no structure information.

Valid Values

PROMPT | GUESS

Behavior

Specify PROMPT to prompt the user.

Specify GUESS for the driver to guess the format of the file.

Default

GUESS

GUI Tab

Advanced tab

Allow Update And Delete

Attribute

AllowUpdateAndDelete (AUD)

Purpose

Allows Update and Delete statements. Because Update and Delete statements cause immediate changes to a text file, only one connection at a time can operate on a file. Each update and delete on a text file can cause significant changes to the file, and performance may be degraded. Consider a more appropriate database form if performance is a significant factor.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), text files are opened exclusively by the current connection.

If set to 0 (Disabled), Update and Delete statements are not allowed.

Default

0 (Disabled)

GUI Tab

Advanced tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI Tab

Advanced tab

Cache Size

Attribute

CacheSize (CSZ)

Purpose

The number of 64 KB blocks the driver uses to cache database records. The larger the number of blocks, the better the performance.

Valid Values

0 | *x*

where:

x

is a positive integer that specifies the number of 64 KB blocks for caching.

Behavior

If set to 0, no records are cached.

If set to *x*, the specified number of 64 KB blocks are set aside for caching. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you are not able to see updates made by other users until you run the Select statement again.

Default

4

GUI Tab

Advanced tab

Century Boundary

Attribute

CenturyBoundary (CB)

Purpose

The cutoff year for century inference when converting two-digit dates to four-digit dates.

Valid Values

xx

where:

xx

is a two-digit number.

Behavior

Two-digit dates that are less than the specified year number are converted to `20xx`. Two-digit dates greater than or equal to the number are converted to `19xx`. For example, using the default value of 20, a date of 19 will be interpreted as 2019 and a date of 21 is interpreted as 1921.

Default

20

GUI Tab

Advanced tab

Column Names in First Line

Attribute

FirstLineNames (FLN)

Purpose

Determines whether the driver looks for column names in the first line of the file.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver looks for column names in the first line of the file.

If set to 0 (Disabled), the driver does not look for column names in the first line of the file.

Notes

- The Column Names in First Line setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Default

0 (Disabled)

GUI Tab

General tab

Data File Extension

Attribute

DataFileExtension (DFE)

Purpose

A one- to three-character file name extension to use for data files.

Valid Values

ext

where:

ext

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

Behavior

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

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

Default

TXT

GUI Tab

Advanced tab

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

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

Valid Values

string

where:

string

is the name of a data source.

Default

None

GUI Tab

General tab

Database Directory

Attribute

Database (DB)

Purpose

The directory that contains the data files.

Valid Values

database_directory

where:

database_directory

is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

Default

None

GUI Tab

General tab

Decimal Symbol

Attribute

DecimalSymbol (CS)

Purpose

The decimal separator used when data is stored.

Valid Values

, | .

Behavior

If set to Comma (,), the driver uses a comma as the decimal separator.

If set to Period (.), the driver uses a period as the decimal separator.

The international decimal symbol (.) must be used in DML statements and parameter buffers.

Default

. (Period)

GUI Tab

Advanced tab

Default Table Type

Attribute

TableType (TT)

Purpose

The type of text file (table) that is used when creating a new table and opening an undefined table.

Valid Values

Comma | Tab | Character | Fixed | Stream

The value chosen determines the type of text used for a table: comma-separated, tab-separated, character-separated, fixed length, or stream.

Notes

- The Default Table Type setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Default

Comma

GUI Tab

General tab

Delimiter Character

Attribute

Delimiter (DC)

Purpose

The character used as a delimiter for character-separated files.

Valid Values

x

where:

x

is any printable character except single quotes, double quotes, or semicolons.

 Note that it is possible to specify a semicolon if you configure the data source using the Windows ODBC Administrator.

Notes

- The Delimiter Character setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Default

, (Comma)

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

Extension List

Attribute

 ExtraExtensions (EE)

Purpose

A comma-separated list of file name extensions for the files that you want returned in addition to the extension specified in the Data File Extension field.

Valid Values

ext | NONE

where:

ext

is a file name extension.

To have files with no extensions returned, specify NONE. For example, if some of your files have the extensions TXT and CSV and others have no extension, specify `TXT,CSV,NONE`.

By default, when an application requests a list of tables, only files that have been defined are returned.

Notes

- You must have also enabled the Files with Matching Extensions option.

Default

None

GUI Tab

Advanced tab

File Open Cache

Attribute

FileOpenCache (FOC)

Purpose

The maximum number of used file handles to cache.

Valid Values

0 | *x*

where:

x

is a positive integer.

Behavior

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default

0 (No File Open Caching)

GUI Tab

Advanced 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

Include Files with Matching Extensions

Attribute

n/a

Purpose

On Windows, enables the driver to return files with a given file name extension in addition to the extension specified through the Data File Extension option. After enabling this option, specify the file name extensions through the Extension List option.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver returns files with the file name extensions specified through the Extension List and Data File Extension options.

If set to 0 (Disabled), the driver returns only files with the file name extension specified through the Data File Extension option.

Default

0 (Disabled)

GUI Tab

Advanced tab

International Sort

Attribute

IntlSort (IS)

Purpose

Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default

0 (Disabled)

GUI Tab

Advanced tab

Rows to Scan

Attribute

ScanRows (SR)

Purpose

The number of rows in a text file that the driver scans to determine the data types in the file.

Valid Values

0 | *x*

where:

x

is a positive integer.

Behavior

If set to 0, all rows in the file are scanned.

If set to *x*, *x* rows are scanned to determine the data types in a file.

Notes

- The Rows to Scan setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Default

25

GUI Tab

Advanced tab

Use Long Qualifiers

Attribute

UseLongQualifiers (ULQ)

Purpose

Determines whether the driver uses long path names.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default

0 (Disabled)

GUI Tab

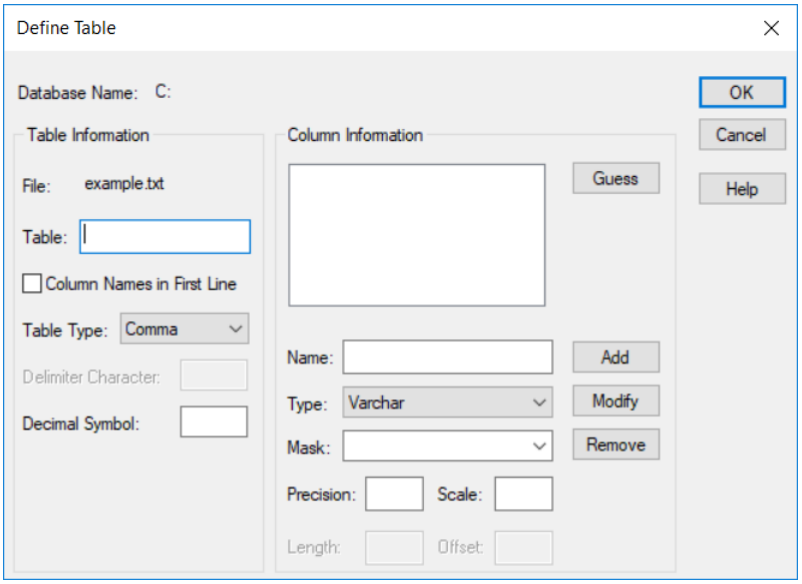
Advanced tab

Defining Table Structure on Windows

Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory (the driver can attempt to guess the names and types of the columns), this feature is extremely useful.

To define the structure of a file:

1. Display the ODBC Text Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.
2. Select the correct file and click **Open** to display the Define Table dialog box.



Database Name: This field displays the name of the database directory that you selected in the Define File dialog box.

File: This field displays the name of the file that you selected in the Define File dialog box.

Table: Type a table name in the Table field. This name specifies the table name associated with the text file you selected earlier. The name can be a maximum of 255 characters and must be unique. This name is returned by SQLTables. By default, it is the file name without its extension (for example, Trc_read).

Column Names in First Line: Select this check box if the first line of the file contains column names; otherwise, do not select this box.

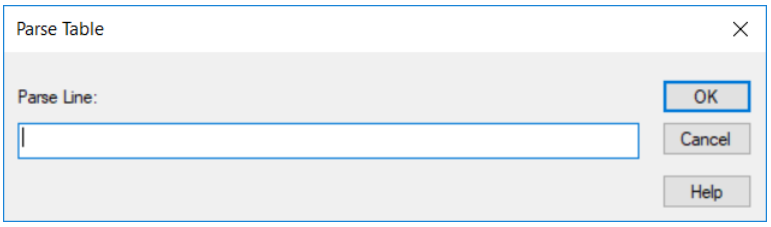
Table Type: Select the type of text file, either comma, tab, fixed, character, or stream.

Delimiter Character: If the table type is Character, type the delimiter used in character-separated files. The value can be any printable character except single and double quotes.

Decimal Symbol: Type the decimal separator used when data is stored. Valid values are a comma or a period. The international decimal symbol (.) must be used in DML statements and parameter buffers.

3. If you specified a comma-separated, tab-separated, or character-separated type in the Table Type field, the Guess/Parse button displays Guess. Click **Guess** to have the driver guess at the column names and display them in the list box of the Column Information pane.

   If you specified a fixed-length or stream type in the Table Type field, the Guess/Parse button displays Parse. Click **Parse** to have the driver display the Parse Table dialog box and define the table columns.



This dialog box displays the first line of the file. You must mark where each field begins and ends by enclosing it in square brackets [ ]. These brackets indicate the position and length of each field value in the record. Click **OK** to close the Parse Table dialog box. The driver will suggest column names in the list box of the Column Information pane.

4. If you do not want the driver to guess or parse, enter values in the following fields to define each column. Click **Add** to add the column name to the Column Information box.

   Name: Type the name of the column.

   Type: Select the data type of the column. If the field type is Date, the Mask field is enabled and you must select a date mask or type one in. See Date Masks on page 235 for more information.

   Mask: Select a date mask. If you selected Date for the Type field, you must select a date mask for the field or type one in. See Date Masks on page 235 for more information.

   Precision: Type the precision of the column. The precision of numeric data types is defined as the maximum number of digits used by the data type of the column. For character types, this is the length in characters of the data. Note that the precision and scale values determine how numeric data is to be returned.

   Scale: Type the scale of the column. The scale of numeric data types is defined as the maximum number of digits to the right of the decimal point. Note that the precision and scale values determine how numeric data is to be returned.

   Length: If you specified a fixed-length table type, type the length, which is the number of bytes the data takes up in storage.

   Offset: If you specified a fixed-length table type, type the offset, which is the number of bytes from the start of the table to the start of the field.

5. To modify an existing column definition, select the column name in the Column Information box. Modify the values for that column name; then, click **Modify**.
6. To delete an existing column definition, select a column name in the Column Information box and click **Remove**.
7. Click **OK** to define the table.

Defining Table Structure on UNIX and Linux

UNIX® Because text files do not all have the same structure, the driver provides the option to define the structure of an existing file. Although defining the structure is not mandatory, because the driver can attempt to guess the names and types of the columns, this feature is extremely useful.

To define the structure of a text file, you create a QETXT.INI file using any plain text editor, such as vi. The file name must be in uppercase. All of the tables you want to define are specified in the QETXT.INI file. When you specify table attributes in QETXT.INI, you override the attributes specified in the system information file (odbc.ini) or in the connection string.

To define the QETXT.INI file:

1. Create a [Defined Tables] section and list all of the tables you are defining. Specify the text file name (in either upper or lowercase, depending on the file) followed by the name you want to give the table, for example:

   ```
   emptext.txt=EMP
   ```

   Table names can be up to 255 characters in length and cannot be the same as another defined table in the database. This name is returned by SQLTables. By default, it is the file name without its extension.

2. For each table listed in the [Defined Tables] section, you must specify the text file (FILE=), the table type (TT=), whether the first line of the file contains column names (FLN=), and the delimiter character (DC=).

   - Specify the text file name. For example:

     ```
     FILE=emptext.txt
     ```

   - To define the table type, specify how the fields are separated (comma, tab, fixed, or character). For example:

     ```
     TT=COMMA
     ```

   - If the table type is CHARACTER, specify the delimiter character. The value can be any printable character except single and double quotes. For example, if the fields are separated by comma:

     ```
     DC=,
     ```

   - Specify whether the first line of the file contains column names, using 1 for yes and 0 for no. For example:

     ```
     FLN=0
     ```

3. Define the fields in the table, beginning with FIELD1. For each field, specify the field name, field type, precision, scale, length, offset (for fixed tables), and date/time mask. See Date Masks on page 235 for information about masks.

   Separate the values with commas. For example, to define two fields:

   ```
   FIELD1=EMP_ID,VARCHAR,6,0,6,0,

   FIELD2=HIRE_DATE,DATE,10,0,10,0,m/d/yy
   ```

4. Save the file as QETXT.INI. The driver looks for this file in the directory specified by the Database attribute in odbc.ini, or in the current directory.

Example of QETXT.INI

The following is an example of a QETXT.INI file. This file defines the structure of the emptext.txt file, which is a sample data file shipped with the DataDirect ODBC Text file.

```
[Defined Tables]
emptext.txt=EMP
[EMP]
FILE=emptext.txt
FLN=1
TT=Comma
FIELD1=FIRST_NAME,VARCHAR,10,0,10,0,
FIELD2=LAST_NAME,VARCHAR,9,0,9,0,
FIELD3=EMP_ID,VARCHAR,6,0,6,0,
```

```
FIELD4=HIRE_DATE,DATE,10,0,10,0,m/d/yy
FIELD5=SALARY,NUMERIC,8,2,8,0,
FIELD6=DEPT,VARCHAR,4,0,4,0,
FIELD7=EXEMPT,VARCHAR,6,0,6,0,
FIELD8=INTERESTS,VARCHAR,136,0,136,0,
```

Date Masks

Date masks tell the driver how a date is stored in a text file. When a value is inserted into a text file, the date is formatted so that it matches the mask. When reading a text file, the driver converts the formatted date into a date data type.

The following table lists the symbols to use when specifying the date mask.

Table 22: Date Masks for Text Driver

Symbol	Description
m	Output the month's number (1–12).
mm	Output a leading zero if the month number is less than 10.
mmm, Mmm, MMM	Output the three-letter abbreviation for the month depending on the case of the Ms (for example, jan, Jan, JAN).
mmmm, Mmmm, MMMM	Output the name of the full month depending on the case of the Ms (for example, january, January, JANUARY).
d	Output the day number (1–31).
dd	Output a leading zero if the day number is less than 10.
ddd, Ddd, DDD	Output the three-letter day abbreviation depending on the case of the Ds (for example, mon, Mon, MON).
dddd, Dddd, DDDD	Output the name of the full day depending on the case of the Ds (for example, monday, Monday, MONDAY).
yy	Output the last two digits of the year.
yyyy	Output the full four digits of the year.
J	Output the Julian value for the date. The Julian value is the number of days since 4712 BC.
\ - . : , (space)	Special characters used to separate the parts of a date.
\	Output the next character. For example, if the mask is mm/dd/yyyy \A\D, the value appears as 10/01/2003 AD in the text file.
"string", 'string'	Output the string in the text file.

The following table shows some example date values, masks, and how the date appears in the text file.

Table 23: Date Mask Examples

Date	Mask	Value
2003-10-01	yyyy-mm-dd	2003-10-01
	m/d/yy	10/1/03
	Ddd, Mmm dd, yyyy	Fri, Oct 01, 2003

Data Types

The following table shows how the text file data types are mapped to the standard ODBC data types.

Table 24: Text Data Types

Text	ODBC
Numeric	SQL_NUMERIC
Date	SQL_TYPE_DATE
Varchar	SQL_VARCHAR

See Retrieving Data Type Information on page 47 for information about retrieving data types.

Select Statement

You use a SQL Select statement to specify the columns and records to be read. All of the Select statement clauses described in SQL Statements for Flat-File Drivers are supported by the Text driver.

Alter Table Statement

The Text driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_namedata_type |
ADD(column_namedata_type [, column_namedata_type]... ) |
DROP[COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept varchar(10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

SQL Support

The driver supports the minimum SQL grammar.

ODBC Conformance Level

The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions. In addition, the SQLSetPos function is supported.

The driver supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll.

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

Number of Connections and Statements Supported

Text files support multiple connections and multiple statements per connection.

8

Drivers Only Available for 32-Bit Platforms

This following sections describe the drivers that are available only in 32-bit versions. See Drivers for 32-Bit and 64-Bit Platforms on page 103 and The Connect XE Drivers on page 327 for information on additional Connect Series drivers.

For details, see the following topics:

- The Btrieve (Pervasive.SQL) Driver
- The dBASE Driver
- The XML Driver

The Btrieve (Pervasive.SQL) Driver

The DataDirect Connect for ODBC Btrieve driver (the Btrieve driver) supports the following versions of Btrieve files:

- Pervasive.SQL
- Btrieve

The driver executes SQL statements directly on Btrieve files.

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

The Btrieve driver is 32-bit only and is supported in the Windows environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Btrieve driver.

Driver Requirements

To access a Btrieve database, you must be using the appropriate client software for the version of the Btrieve database to which you are connecting:

Database Versions	Client Names
Pervasive.SQL 8.5	Pervasive.SQL 8.5 client software
Pervasive.SQL 2000	Pervasive.SQL 2000 client software
Pervasive.SQL 7.0	Pervasive.SQL 7.0 client software
Btrieve 6.15 for Windows 9*x*	Btrieve Developer's Kit orBtrieve WorkStation Client Engine
Btrieve 6.15 for Windows NT	Btrieve Developer's Kit, Btrieve WorkStation Client Engine, or Btrieve Client/Server Database Engine

Note: The Btrieve driver may experience problems if the Btrieve Microkernel Engine's communication buffer size is smaller than that of the Btrieve driver's Array Size option. You can increase the communication buffer size with the Pervasive Software Setup Utility, or you can decrease the value of Array Size option through the ODBC Btrieve Driver setup dialog box or through the ArraySize connection string attribute.

Before you attempt to access Btrieve files, you must incorporate existing Btrieve files into a Scalable SQL database. See Managing Databases on page 240 for information about Scalable SQL databases.

Managing Databases

If you already use Scalable SQL, the Btrieve driver can access your Scalable SQL databases directly. If not, your Btrieve files must be incorporated into a Scalable SQL database.

A Scalable SQL database is composed of data files that contain your records and data dictionary files that describe the database. The data files are Btrieve files. The data dictionary files are special Btrieve files that contain descriptions of the data files, views, fields, and indexes in your database.

All Btrieve files in a Scalable SQL database must reside in the same directory. In addition to the Btrieve data files, the three data dictionary files (FILE.DDF, FIELD.DDF, and INDEX.DDF) also must be in the directory.

Incorporating a Btrieve file into a Scalable SQL database does not change the Btrieve file in any way. You can continue to access the file directly with any existing Btrieve application.

Transactions

The Btrieve driver supports *transactions*. A transaction is a series of database changes that is treated as a single unit. In applications that so not use transactions, the Btrieve driver immediately executes Insert, Update, and Delete statements on the database files and the changes are automatically committed when the SQL statement is executed. You cannot undo these changes. In applications that use transactions, the Btrieve driver holds inserts, updates, and deletes until you issue a Commit or Rollback. A Commit saves the changes to the database file; a Rollback undoes the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

To use the Btrieve driver's transaction processing capabilities, consult the Pervasive documentation.

Configuring and Connecting to Data Sources (Btrieve)

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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. See Data Source Configuration through a GUI (Btrieve) on page 241 for details.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 244 and Connection Option Descriptions on page 245 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration through a GUI (Btrieve)

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.

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

To configure a Btrieve data source:

1. Start the ODBC Administrator by selecting its icon from the DataDirect Connect program group; then, select a tab:

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

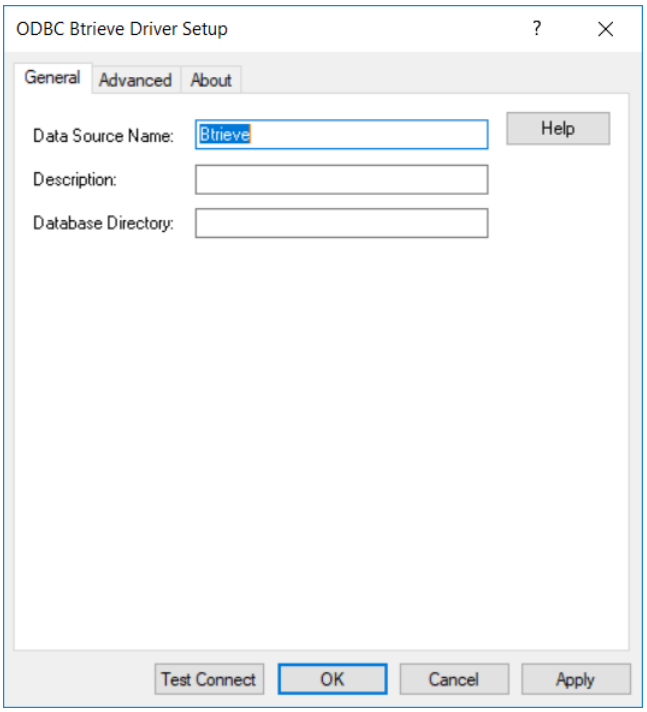
   If you are configuring a new user data source, click **Add** on the User DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

   - **System DSN**: To configure a new system data source, click **Add** on the System DSN tab 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 name on the File DSN tab and click **Configure** to display the driver Setup dialog box.

     If you are configuring a new file data source, click **Add** on the File DSN tab to display a list of installed drivers. Select the driver and click **Next**. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

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

Figure 20: General tab



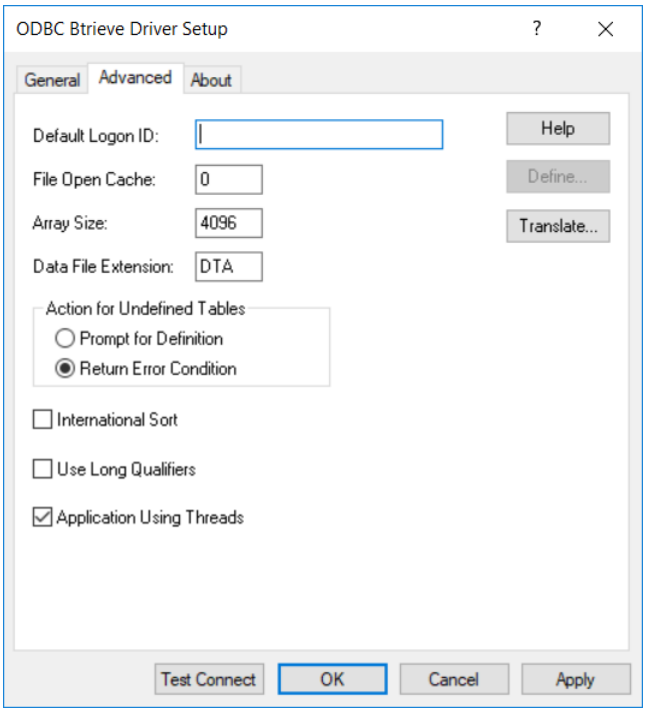
Note: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

2. On this tab, provide values for the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 248	None
Description on page 250	None
Database Directory on page 249	None

3. Optionally, click the **Advanced** tab to specify data source settings.

Figure 21: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Default Logon ID on page 249	None
File Open Cache on page 250	0
Array Size on page 247	4096
Data File Extension on page 248	DTA
Action for Undefined Tables on page 246	Return Error Condition
International Sort on page 251	Disabled
Use Long Qualifiers on page 252	Disabled
Application Using Threads on page 247	Enabled

Define: Click **Define** to define table structure as described in Defining Table Structure on page 253.

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

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

4. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

   - If the driver can connect, it releases the connection and displays a `Connection established!` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message.

   Verify that all required client software is properly installed. If it is not, you will see the message:

   ```
   Specified driver could not be loaded due to system error [xxx].
   ```

   Click **OK**.

5. Click **OK** or **Cancel**. If 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.

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 on page 245 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 Btrieve is:

```
DSN=BTRIEVE FILES;DB=J:\Btrvdata
```

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

```
FILEDSN=Btrieve.dsn;DB=J:\Btrvdata
```

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

```
DRIVER={DataDirect 7.1 Btrieve};DB=J:\Btrvdata;UID=JOHN;PWD=XYZZY
```

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 on page 245 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 Btrieve is:

```
DSN=BTRIEVE FILES;DB=J:\Btrvdata
```

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

```
FILEDSN=Btrieve.dsn;DB=J:\Btrvdata
```

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

```
DRIVER={DataDirect 7.1 Btrieve};DB=J:\Btrvdata;UID=JOHN;PWD=XYZZY
```

Connection Option Descriptions

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

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

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

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

The following table lists the connection string attributes supported by the Btrieve driver.

Table 25: Btrieve Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	4096
Database (DB)	None
DataFileExtension (DFE)	DTA
DataSourceName (DSN)	None
Description (n/a)	None
FileOpenCache (FOC)	0 (No File Open Caching)
IntlSort (IS)	0 (Disabled)
LogonID (UID)	None
Password (PWD)	None
UndefinedTable (UT)	Error
UseLongQualifiers (ULQ)	0 (Disabled)

Action for Undefined Tables

Attribute

UndefinedTable (UT)

Purpose

Determines whether the driver prompts the user when it encounters a table for which it has no structure information.

Valid Values

PROMPT | ERROR

Specify PROMPT to prompt the user.

Specify ERROR to return an error.

Default

ERROR (driver returns an error)

GUI Tab

Advanced tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI Tab

Advanced tab

Array Size

Attribute

ArraySize (AS)

Purpose

The number of bytes in the array. This connection option enables the driver to retrieve an array of records from the Btrieve database and, in most cases, results in improved performance for the application.

Valid Values

A positive integer from 1 to 65535

Default

4096

GUI Tab

Advanced tab

Data File Extension

Attribute

DataFileExtension (DFE)

Purpose

A one- to three-character file name extension to use for data files.

Valid Values

ext

where:

ext

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

Behavior

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

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

Default

DTA

GUI Tab

Advanced tab

Data Source Name

Attribute

DataSourceName (DSN)

Purpose

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

Valid Values

string

where:

string

is the name of a data source.

Default

None

GUI Tab

General tab

Database Directory

Attribute

Database (DB)

Purpose

The directory that contains the data files.

Valid Values

database_directory

where:

database_directory

is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

This includes both Btrieve files and the data dictionary files (.DDF). Data dictionary files describe the structure of Btrieve data.

Default

None

GUI Tab

General tab

Default Logon ID

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

Advanced 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

File Open Cache

Attribute

FileOpenCache (FOC)

Purpose

The maximum number of used file handles to cache.

Valid Values

0 | *x*

where:

x

is a positive integer.

Behavior

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default

0 (No File Open Caching)

GUI Tab

Advanced tab

International Sort

Attribute

IntlSort (IS)

Purpose

Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default

0 (Disabled)

GUI Tab

Advanced tab

Password

Attribute

Password (PWD)

Purpose

The password that you must enter if your Scalable SQL data dictionary files have security restrictions set. The Password option cannot be specified through the Administrator GUI.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Use Long Qualifiers

Attribute

UseLongQualifiers (ULQ)

Purpose

Determines whether the driver uses long path names.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default

0 (Disabled)

GUI Tab

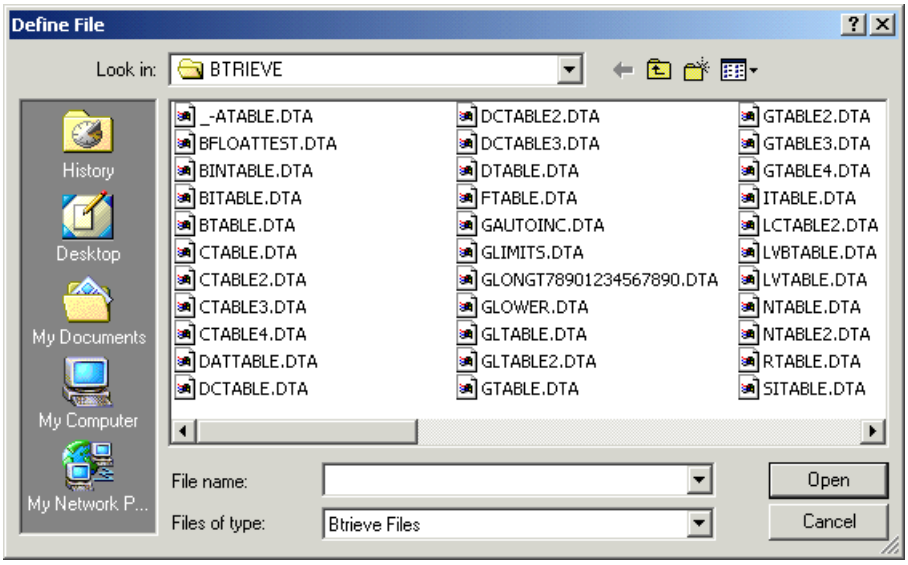
Advanced tab

Defining Table Structure

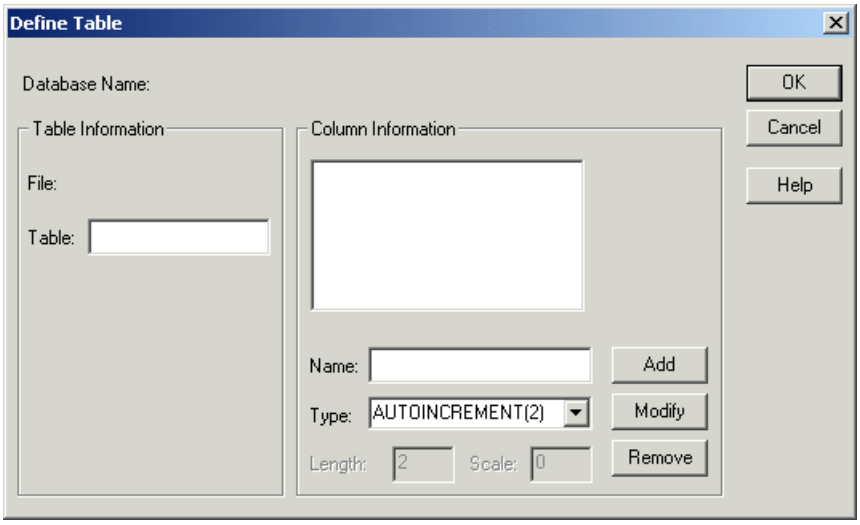
Because Btrieve does not store any column information in the data file, you may need to define its structure. Tables created by the Btrieve driver or by Scalable SQL will not require this. Utilities are also available from Pervasive Software that will perform this operation.

To define the structure of a file:

1. Display the ODBC Btrieve Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.



2. In the Define File dialog box, select the file you want to define and click **Open** to display the Define Table dialog box.



Database Name: This field displays the directory in which the file you selected in the Define File dialog box is located.

File: This field displays the name of the file that you selected in the Define File dialog box.

Table: Type the name of the table to be returned by SQLTables. The name can be a maximum of 20 characters and cannot be the same as another defined table in the database. This field is required.

3. Specify values in the following fields to define each column. Click **Add** to add the column name to the list box.

   Name: Type the name of the column.

   Type: Select the data type of the column.

   Length: Type the length of the column, if applicable.

   Scale: Type the scale of the column, if applicable.

4. To modify an existing column definition, select the column name in the list box. Modify the values for that column name; then, click **Modify**.
5. To delete an existing column definition, select a column name in the list box and click **Remove**.
6. Click **OK** to define the table.

Data Types

The following table shows how the Btrieve data types map to the standard ODBC data types. The Btrieve data types are used when you incorporate Btrieve files into a Scalable SQL database.

Table 26: Btrieve Data Types

Btrieve	ODBC
Autoincrement(2)	SQL_SMALLINT
Autoincrement(4)	SQL_INTEGER
Bfloat(4)	SQL_REAL
Bfloat(8)	SQL_DOUBLE
Bit	SQL_BIT
Blob	SQL_LONGVARGBINARY
Char	SQL_CHAR
Currency	SQL_DECIMAL
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Float(4)	SQL_REAL
Float(8)	SQL_DOUBLE
Integer(1)	SQL_TINYINT

Btrieve	ODBC
Integer(2)	SQL_SMALLINT
Integer(4)	SQL_INTEGER
Integer(8)	SQL_BIGINT
Logical(1)	SQL_BIT
Logical(2)	SQL_BIT
Lstring	SQL_VARCHAR
Money	SQL_DECIMAL
Note	SQL_LONGVARCHAR
Numeric	SQL_NUMERIC
Numericsts	SQL_NUMERIC
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Unsigned(1)	SQL_TINYINT
Unsigned(8)	SQL_BIGINT
Zstring	SQL_VARCHAR

See Retrieving Data Type Information on page 47 for information about retrieving data types.

Indexes

Note: If you define an index using the Btrieve driver, the index will not have the restrictions discussed here.

For query optimization, the Btrieve driver does not use the following:

- Indexes containing all-segment-null keys or any-segment-null keys.
- Any index key that is marked case-insensitive.
- Any index keys where the data type of the index key does not match the data type of the field. The one exception is if the index key is declared as an unsigned integer and the field in the file is declared as signed integer, or vice versa, then the driver assumes the field contains only unsigned quantities and uses the index. Note that this can lead to incorrect results if the field in fact does contain signed quantities.

The Btrieve driver only uses an alternate-collating-sequence (ASC) index key for equality lookups. Additionally, if an ASC key is part of a segmented index, the other index segments are not used for query optimization unless the Where clause contains an equality condition for the ASC key.

Column Names

Column names in SQL statements (such as Select and Insert) can be up to 20 characters long. If column names are in all lowercase, a combination of upper and lowercase, contain blank spaces, or are reserved words, they must be surrounded by the grave character (`) (ASCII 96). For example:

```
SELECT `name` FROM emp
```

Select Statement

You use the SQL Select statement to specify the columns and records to be read. Btrieve Select statements support all the Select statement clauses described in SQL Statements for Flat-File Drivers. This section describes the information that is specific to Btrieve.

Rowid Pseudo-Column

Each Btrieve record contains a special column named Rowid. This field contains a unique number that indicates the record's sequence in the database. You can use Rowid in Where and Select clauses.

Rowid is particularly useful when you are updating records. You can retrieve the Rowid of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM emp
```

Then, you can use the Rowid of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=21
```

The fastest way of updating a single row is to use a Where clause with the Rowid. You cannot update the Rowid column.

Select statements that use the Rowid pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21    //fast search
SELECT * FROM emp WHERE rowid <=25  //full table scan
```

Alter Table Statement

The Btrieve driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_namedata_type | ADD (column_namedata_type [,
column_namedata_type]...)
| DROP [COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept char 10)
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

Create and Drop Index Statements

The Btrieve driver supports SQL statements to create and delete indexes.

Create Index

The Create Index statement for Btrieve files has the form:

```
CREATE [UNIQUE] INDEX index_name ON table_name ([field_name [ASC | DESC] [,field_name
[ASC | DESC]]...)
```

Unique means that Btrieve does not let you insert two records with the same index values.

index_name is the name of the index.

table_name is the name of the table on which the index is to be created.

ASC tells Btrieve to create the index in ascending order. DESC tells Btrieve to create the index in descending order. By default, indexes are created in ascending order. For example:

```
CREATE INDEX lname ON emp (last_name)
```

Drop Index

The form of the Drop Index statement is:

```
DROP INDEX table_name.index_name
```

table_name is the name of the table from which the index is to be dropped.

index_name is the name of the index.

For example:

```
DROP INDEX emp.lname
```

Isolation and Lock Levels Supported

Btrieve supports isolation level 1 (read committed) only. Btrieve supports record-level locking.

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

SQL Support

The driver supports the minimum SQL grammar with several core extensions.

ODBC Conformance Level

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

In addition, the following function is supported: SQLSetPos.

The driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll.

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

Number of Connections and Statements Supported

Btrieve files support a single connection and multiple statements per connection.

The dBASE Driver

The DataDirect Connect for ODBC dBASE driver (the dBASE driver) supports the following file types:

- dBASE
- Clipper
- FoxPro
- FoxPro database container (DBC)

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

The dBASE driver runs the SQL statements directly on dBASE- and FoxPro-compatible files. You do not need to own dBASE or FoxPro products to access these files. The dBASE driver cannot access files that are larger than 2 GB.

The dBASE driver is 32-bit only and is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the dBASE driver.

Driver Requirements

There are no client requirements for the dBASE driver.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 265 and Connection Option Descriptions on page 265 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX/Linux odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Connection Option Descriptions lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (dBase)

 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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

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

To configure a dBASE data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

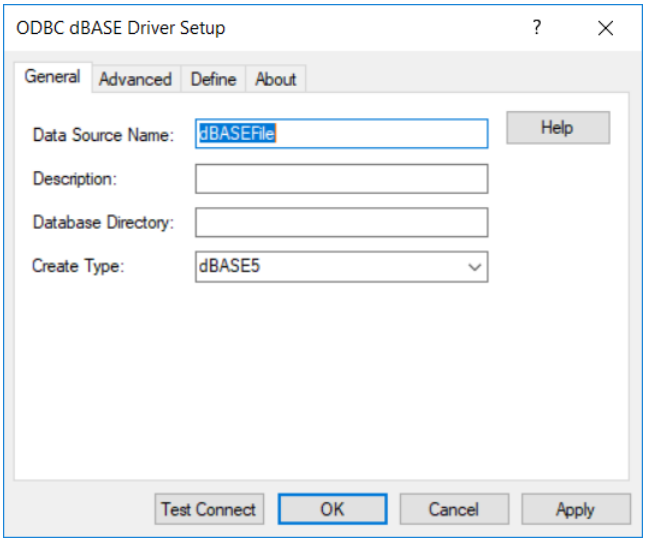
  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **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.

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

Figure 22: General tab



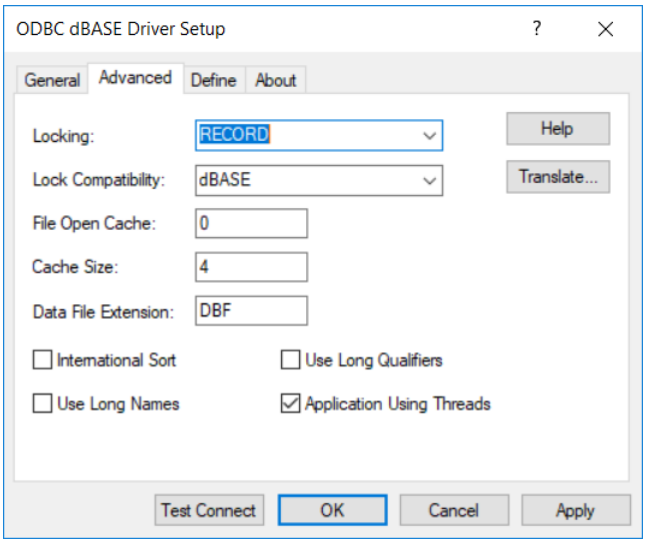
Note: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3. On this tab, provide values for the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 269	None
Description on page 271	None
Database Directory on page 270	None
Create Type [dBASE] on page 268	dBASE5

4. Optionally, click the **Advanced** tab to specify data source settings.

Figure 23: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Locking on page 274	RECORD
Lock Compatibility on page 274	dBASE
File Open Cache on page 272	0
Cache Size on page 267	4
Data File Extension on page 268	DBF
International Sort on page 273	Disabled
Use Long Names on page 275	Disabled
Use Long Qualifiers on page 275	Disabled
Application Using Threads on page 266	Enabled
IANAAppCodePage on page 272 UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

5. If you use index files that have different names than their corresponding data files and you have not defined this association, click the **Define** tab. See Defining Index Attributes on Windows on page 276 for step-by-step instructions.

6. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.
   - If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.
7. Click **OK** or **Cancel**. If 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.

Data Source Configuration through a GUI (FoxPro)


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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

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

To configure a FoxPro 3.0 database container data source :

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

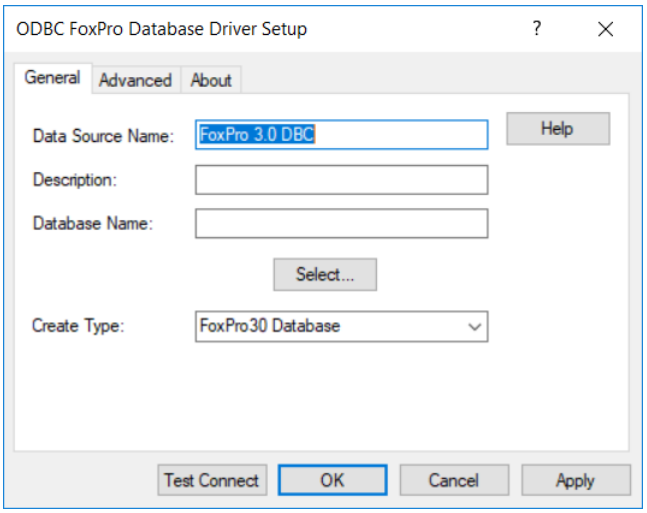
     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** 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.

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

Figure 24: General tab



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

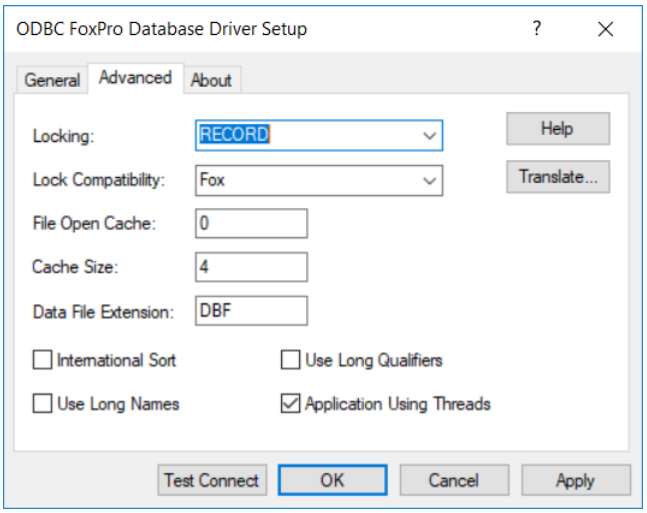
3. On this tab, provide values for the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 269	None
Description on page 271	None
Database Name on page 270	None
Create Type [FoxPro] on page 268	FoxPro30 Database

Click **Select** to choose the directory and .DBC file that you want to use.

4. Optionally, click the **Advanced** tab to specify data source settings.

Figure 25: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Locking on page 274	RECORD
Lock Compatibility on page 274	Fox
File Open Cache on page 272	0
Cache Size on page 267	4
Data File Extension on page 268	DBF
International Sort on page 273	Disabled
Use Long Names on page 275	Disabled
Use Long Qualifiers on page 275	Disabled
Application Using Threads on page 266	Enabled
IANAAppCodePage on page 272 UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

5. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

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

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

6. Click **OK** or **Cancel**. If 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.

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]...]
```

The following table 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 dBASE is:

```
DSN=DBASE FILES;LCK=NONE;IS=0
```

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

```
FILEDSN=DBASE.dsn;LCK=NONE;IS=0
```

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

```
DRIVER={DataDirect 7.1 dBASEFile (*.dbf)};DB=C:\DBASE;CT=dBASE5
```

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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the dBASE driver.

Table 27: dBASE Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
CacheSize (CSZ)	4
CreateType (CT) [dBASE]	dBASE5
CreateType (CT)) [FoxPro]	FoxPro30 Database
Database (DB) [dBASE]	None
Database (DB) [FoxPro]	None
DataFileExtension (DFE)	DBF
DataSourceName (DSN)	None
Description (n/a)	None
ExtensionCase (EC)	UPPER
FileOpenCache (FOC)	0 (no file open caching)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntlSort (IS)	0 (Disabled)
LockCompatibility (LCOMP)	dBASE
Locking (LCK)	RECORD
UseLongNames (ULN)	0 (Disabled)
UseLongQualifiers (ULQ)	0 (Disabled)

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Cache Size

Attribute

CacheSize (CSZ)

Purpose

The number of 64 KB blocks the driver uses to cache database records. The larger the number of blocks, the better the performance.

Valid Values

0 | *x*

where:

x

is a positive integer that specifies the number of 64 KB blocks for caching.

Behavior

If set to 0, no records are cached.

If set to *x*, the specified number of 64 KB blocks are set aside for caching. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you are not able to see updates made by other users until you run the Select statement again.

Default

4

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Create Type [dBASE]

Attribute

CreateType (CTS) [dBASE]

Purpose

The type of table or index to be created on a Create Table or Create Index statement.

Valid Values

dBASE4 | dBASE5 | Clipper | FoxPro25 | FoxPro30

Default

dBASE5

GUI tab

General tab [dBASE]

Create Type [FoxPro]

Attribute

CreateType (CT)) [FoxPro]

Purpose

The type of table or index to be created on a Create Table or Create Index statement.

Valid Value

FoxPro30 Database

Default

FoxPro30 Database

GUI tab

General tab [FoxPro]

Data File Extension

Attribute

DataFileExtension (DFE)

Purpose

A one- to three-character file name extension to use for data files.

Valid Values

ext

where:

ext

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

Behavior

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

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

The file extension cannot be one the driver already uses, such as MDX or CDX.

Default

DBF

GUI tab

General tab [dBASE]

General tab [FoxPro]

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 [dBASE]

General tab [FoxPro]

Database Directory

Attribute

Database (DB) [dBASE]

Purpose

The directory that contains the data files.

Valid Values

database_directory

where:

database_directory

is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

Default

None

GUI Tab

General tab [dBASE]

Database Name

Attribute

Database (DB) [FoxPro]

Purpose

The directory that contains the database container (.DBC) files.

Valid Values

database_directory

where:

database_directory

is the full path name of the directory and .DBC file that you want to use.

Default

None

GUI Tab

General tab [FoxPro]

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 [dBASE]

General tab [FoxPro]

Extension Case

Attribute

ExtensionCase (EC)

Purpose

This option determines whether uppercase or lowercase file extensions are accepted.

Valid Values

LOWER | UPPER

Behavior

When set to UPPER, uppercase extensions are accepted.

When set to LOWER, lowercase extensions are accepted.

Default

UPPER

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

File Open Cache

Attribute

FileOpenCache (FOC)

Purpose

The maximum number of used file handles to cache.

Valid Values

0 | *x*

where:

x

is a positive integer.

Behavior

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default

0 (No File Open Caching)

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

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 [dBASE]

Advanced tab [FoxPro]

International Sort

Attribute

IntlSort (IS)

Purpose

Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default

0 (Disabled)

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Lock Compatibility

Attribute

LockCompatibility (LCOMP)

Purpose

The locking scheme the driver uses when locking records.

Valid Values

Clipper | dBASE | Fox | Q+E | Q+EVirtual

- Clipper specifies Clipper-compatible locking.
- dBASE specifies Borland-compatible locking.
- Fox specifies FoxPro-compatible locking.
- Q+E specifies that locks be placed on the actual bytes occupied by the record. Only applications that use the dBASE driver can read and write to the database. Other applications are locked out of the table completely (they cannot even read other records). This locking is compatible with earlier versions of Q+E products.
- Q+EVirtual specifies that locks be placed on bytes beyond the physical end-of-file. Q+EVirtual is the same as Q+E except that other applications can open the table and read the data.

The advantage of using a Q+E locking scheme over dBASE locking is that, on Inserts and Updates, Q+E locks only individual index tags, while dBASE locks the entire index. The following values determine locking support as described:

If you are accessing a table with an application that uses the dBASE driver, your locking scheme does not have to match the Create Type. If you access a table with two applications, however, and only one uses the dBASE driver, set your locking scheme to match the other application. For example, you do not have to set this value to Fox to work with a FoxPro table. But if you are using a FoxPro application simultaneously with an application using the dBASE driver on the same set of tables, set this value to Fox to ensure that your data does not become corrupted.

Default

dBASE

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Locking

Attribute

Locking (LCK)

Purpose

The level of locking for the database file.

Valid Values

NONE | RECORD | FILE

- NONE offers the best performance, but is intended only for single-user environments. See Locking on page 282 for details.
- RECORD locks only the records affected by the statement.
- FILE locks all of the records in the table.

Default

RECORD

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Use Long Names

Attribute

UseLongNames (ULN)

Purpose

Specifies whether to use long file names as table names.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver uses long file names as table names. The maximum table name length is specific to the environment in which you are running.

If set to 0 (Disabled), the driver does not long file names as table names.

Default

0 (Disabled)

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

Use Long Qualifiers

Attribute

UseLongQualifiers (ULQ)

Purpose

Determines whether the driver uses long path names.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default

0 (Disabled)

GUI tab

Advanced tab [dBASE]

Advanced tab [FoxPro]

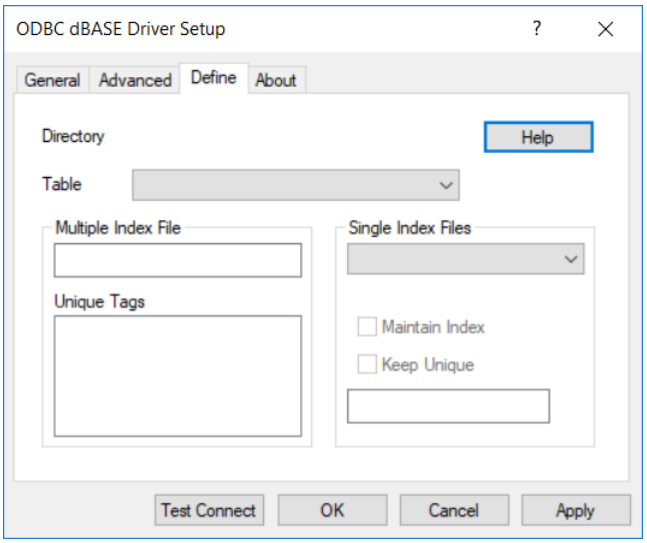
Defining Index Attributes on Windows

The Define tab of the ODBC dBASE Driver Setup dialog box allows you to define the attributes of index files. With the exception of Clipper, the family of databases that includes dBASE and FoxPro uses a multiple index file associated with a particular table (database file). This index file has a .MDX or .CDX extension and is automatically maintained by the driver. Tags within this index can be marked as unique.

Clipper uses single index files that are not automatically associated with a particular table. You can choose to have the driver maintain an index and choose whether or not the index is unique.

To define index file attributes:

1. Display the ODBC dBASE Driver Setup dialog box.
2. Click the **Define** tab.



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

Table: Type or select the name of the table that contains the database information.

Multiple Index File: This field displays the name of any multiple index file (with a .CDX extension or .MDX extension) associated with the table you selected. This index file cannot be marked as unique, but tags within it can be.

Unique Tags: This field displays tags associated with the multiple index file. To mark tags as unique, click each one; each one remains selected until you click it again.

Single Index Files: The Single Index Files group is active only if you have selected a Clipper table.

Select the file from the drop-down list to define the attributes of a single index file.

Maintain Index: Select this check box to associate the specified single index file with the selected table.

Keep Unique: Select this check box to specify that the single index file is unique.

3. Click **OK** or **Cancel**. If 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.

Defining Index Attributes on UNIX and Linux

UNIX® Index files for dBASE contain index tags for each index that exists for a database file. These index tags can be marked as unique, that is, the driver will ensure that no duplicate values exist for the columns that define the index tag. The unique attribute is not natively supported by the dBASE or FoxPro products. The enforcement and recognition of the unique attribute is an extension of the dBASE driver. The driver must be notified that index tags are unique. No configuration is needed for unique indexes that were created using the DataDirect Connect for ODBC dBASE driver. When using files that were not created with the dBASE driver, you must define unique index tags as outlined in the following procedure.

In the directory where the database and index files are located, use any text editor, such as vi, to define or edit the QEDBF.INI as follows:

1. Create a [*filename*] section where *filename* is the name of the database file. This entry is case-sensitive and the file extension must be included, for example, [accts.dbf].
2. In the [*filename*] section, specify the number of unique indexes on the file (NUMUNIQUE=) and the index specifications (UNIQUE#=*index_filename*,*index_tag*). The *index_tag* can be determined by calling the ODBC function SQLStatistics and examining the INDEX_NAME result column.

For example, to define two unique indexes on the accts.dbf database file, the QEDBF.INI would be defined as:

```
[accts.dbf]
NUMUNIQUE=2
UNIQUE0=accts.mdx,ACCT_NAME
UNIQUE1=accts.mdx,ACCT_ID
```

Data Types

The following table shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement.

The following table shows how the additional FoxPro 3.0 data types map to the ODBC data types.

NOTE: A few products can create dBASE files with numbers that do not conform to the precision and scale of the Number column. For example, these products can store 100000 in a column declared as NUMBER(5,2). When this occurs, the dBASE driver displays error 1244, `Unsupported decimal format`. To remedy this situation, multiply the nonconforming column by 1, which converts it to the Float data type. For example:

```
SELECT BADCOL * 1 FROM BADFILE
```

`BADCOL * 1` is evaluated as an expression and is returned as a float value.

Table 28: dBASE Data Types

dBASE	ODBC
Binary[7]	SQL_LONGVARBINARY
Char[8]	SQL_CHAR
Date	SQL_TYPE_DATE
Float[9]	SQL_DECIMAL
General[10]	SQL_LONGVARBINARY
Logical	SQL_BIT
Memo	SQL_LONGVARCHAR
Numeric	SQL_DECIMAL

Table 29: Additional FoxPro 3.0 Data Types

FoxPro 3.0	ODBC
Character (binary)	SQL_CHAR
Currency	SQL_DOUBLE
Datetime	SQL_TYPE_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
Memo (binary)	SQL_LONGVARBINARY

See Retrieving Data Type Information on page 47 for information about retrieving data types.

7 dBASE V only.
8 254 characters maximum (1024 for Clipper).
9 dBASE IV and V only.
10 FoxPro and dBASE V only.

Column Names

Column names in SQL statements (such as Select and Insert, for example) can be up to ten characters long. A column name can contain alphanumeric characters and the hyphen character (-). The first character must be a letter (a through z).

Select Statement

You use a SQL Select statement to specify the columns and records to be read. All of the Select statement clauses described in SQL Statements for Flat-File Drivers are supported by dBASE Select statements. This section describes the information that is specific to dBASE, which is Rowid.

Rowid Pseudo-Column

Each dBASE record contains a special column named Rowid. This field contains a unique number that indicates the record's sequence in the database. For example, a table that contains 50 records has Rowid values from 1 to 50 (if no records are marked deleted). You can use Rowid in Where and Select clauses.

Rowid is particularly useful when you are updating records. You can retrieve the Rowid of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM emp
```

Then, you can use the Rowid of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=21
```

The fastest way of updating a single row is to use a Where clause with the Rowid. You cannot update the Rowid column.

Select statements that use the Rowid pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21    //fast search
SELECT * FROM emp WHERE rowid <=25  //full table scan
```

Alter Table Statement

The dBASE driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_namedata_type |
ADD(column_namedata_type [, column_namedata_type]... ) |
DROP[COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept char (10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails if you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

Create and Drop Index Statements

The dBASE driver supports SQL statements to create and delete indexes.

Create Index

The type of index you create is determined by the value of the CreateType attribute, which you set in the driver Setup dialog box (for UNIX and Linux, edit the system information file) or as a connection string attribute. The index can be:

- dBASE IV or V (.MDX)
- Clipper (.NTX)
- FoxPro (.CDX)

The syntax for creating an index is:

```
CREATE [UNIQUE] INDEX index_name ON base_table_name
(field_name [ASC | DESC] [,field_name [ASC | DESC]]...)
```

Unique means that the driver creates an ANSI-style unique index over the column and ensures uniqueness of the keys. Use of unique indexes improves performance. ANSI-style unique indexes are different from dBASE-style unique indexes. With ANSI-style unique indexes, you receive an error message when you try to insert a duplicate value into an indexed field. With dBASE-style unique indexes, you do not see an error message when you insert a duplicate value into an indexed field. This is because only one key is inserted in the index file.

index_name is the name of the index file. For FoxPro and dBASE IV or V, this is a tag, which is required to identify the indexes in an index file. Each index for a table must have a unique name.

base_table_name is the name of the database file whose index is to be created. The .DBF extension is not required; the driver automatically adds it if it is not present. By default, dBASE IV or V index files are named *base_table_name*.MDX and FoxPro indexes are named *base_table_name*.CDX.

field_name is a name of a column in the dBASE table. You can substitute a valid dBASE-style index expression for the list of field names.

ASC tells dBASE to create the index in ascending order. DESC tells dBASE to create the index in descending order. By default, indexes are created in ascending order. You cannot specify both ASC and DESC orders within a single Create Index statement. For example, the following statement is invalid:

```
CREATE INDEX emp_i ON emp (last_name ASC, emp_id DESC)
```

The following table shows the attributes of the different index files supported by the dBASE driver. For each type supported, it provides the following details:

- Whether dBASE-style unique indexes are supported

- Whether descending order is supported
- The maximum size supported for key columns
- The maximum size supported for the column specification in the Create Index statement
- Whether production/structural indexes are supported

Table 30: dBASE-Compatible Index Summary

Create Type.Extension	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Specification	Production/Structural Indexes	Supports FOR Expressions
dBASE IV, V .MDX	Yes	Yes	100	220	Yes	Yes
Clipper .NTX	Yes	Yes	250	255	No	Yes
FoxPro .IDX[11]	Yes	Yes	240	255	No	Yes
FoxPro .CDX	Yes	Yes	240	255	Yes	Yes

Drop Index

The syntax for dropping an index is as follows:

```
DROP INDEX table_name.index_name
```

table_name is the name of the dBASE file without the extension.

For FoxPro and dBASE IV or V, *index_name* is the tag. Otherwise, *index_name* is the name of the index file without the extension.

To drop the index EMPHIRE.MDX, issue the following statement:

```
DROP INDEX emp.emphire
```

Pack Statement

When records are deleted from a dBASE file, they are not removed from the file. Instead, they are marked as having been deleted. Also, when memo fields are updated, space may be wasted in the files. To remove the deleted records and free the unused space from updated memo fields, you must use the Pack statement. It has the following form:

```
PACK filename
```

filename is the name of the dBASE file to be packed. The .DBF extension is not required; the driver automatically adds the extension if it is not present. For example:

```
PACK emp
```

You cannot pack a file that is opened by another user, and you cannot use the Pack statement in manual commit mode.

[11] Compact IDX indexes have the same internal structure as a tag in a CDX file. These indexes can be created if the IDX extension is included with the index name in the Create Index statement.

For the specified file, the Pack statement performs the following actions:

- Removes all deleted records from the file
- Compresses unused space in the memo file (.DBT or .FPT)
- Removes the entries for all deleted records from .CDX and .MDX files having the same name as the file

SQL Statements for FoxPro 3.0 Database Containers

The FoxPro DBC driver supports four additional SQL statements:

- Create Database
- Add Table
- Remove Table
- Use

To create a new FoxPro 3.0 database container, use:

```
CREATE DATABASE database_name
```

To add an existing table to the database container, use:

```
ADD TABLE table_name
```

To remove a table from the database container (not delete the table, but unlink it from the database container), use:

```
REMOVE TABLE table_name
```

To set the current database container to an existing database container, use:

```
USE database_name
```

To add or delete columns from a table in a database container, use the Alter Table statement (see Alter Table Statement on page 279).

Locking

With the dBASE driver, you can build and run applications that share dBASE database files on a network. Whenever more than one user is running an application that accesses a shared database file, the applications should lock the records that are being changed. Locking a record prevents other users from locking, updating, or deleting the record.

Levels of Database Locking

The dBASE driver supports three levels of database locking: NONE, RECORD, and FILE. You can set these levels in:

- The connection string (LCK=)
- The Setup dialog box

No locking offers the best performance, but is intended only for single-user environments.

With record or file locking, the system locks the database files during Insert, Update, Delete, or Select...For Update statements. The locks are released when the user commits the transaction. The locks prevent other users from modifying the locked objects, but they do not lock out readers.

With record locking, only records affected by the statement are locked. Record locking provides better concurrency with other users who also want to modify the database file.

With file locking, all the records in the database file are locked. File locking has lower overhead and may work better if records are modified infrequently, if records are modified primarily by one user, or if a large number of records are modified.

Limit on Number of Locks

There is a limit on the number of locks that can be placed on a file. If you are accessing a dBASE file from a server, the limit depends on the server (refer to your server documentation).

If you are accessing a dBASE file locally, the limit depends on the buffer space allocated when SHARE.EXE was loaded (refer to your DOS documentation). If you are exceeding the number of locks available, you may want to switch to file locking.

How Transactions Affect Record Locks

When an Update or Delete statement is run, the driver locks the records affected by that statement. The locks are released after the driver commits the changes. Under manual commit mode, the locks are held until the application commits the transaction. Under autocommit mode, the locks are held until the statement is run.

When a Select...For Update statement is run, the driver locks a record only when the record is fetched. If the record is updated, the driver holds the lock until the changes are committed. Otherwise, the lock is released when the next record is fetched.

Isolation and Lock Levels Supported

dBASE supports isolation level 1 (read committed). It supports both file-level and record-level locking.

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

SQL Support

The driver supports the minimum SQL grammar.

ODBC Conformance Level

The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions. In addition, the SQLSetPos function is supported.

The driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll.

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

Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

The XML Driver

The DataDirect Connect for ODBC XML driver (the XML driver) supports:

Tabular- and hierarchical-formatted XML documents that can be accessed from either a local file system, a web server, or a web service. The three main types of tabular-formatted files that the driver supports are Microsoft Data Islands, ADO 2.5 persisted files, and DataDirect Format.

See Supported Tabular Formats for XML Documents on page 285 for more details.

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

The XML driver is 32-bit only and is supported in the Windows environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

The XML driver includes a SQL Engine that provides ANSI SQL-92 support. The following table lists the SQL statements that the driver supports for the different types of file formats.

File Format	Select	Create/Drop	Insert	Update	Delete
Tabular, Microsoft Data Islands	X	X	X	X	X
Tabular, ADO 2.5 Persisted	X	X	X	X	X
Tabular, DataDirect	X	X	X	X	X
Tabular, other formats	X		X	X	X
Hierarchical	X				

See SQL Support on page 317 for more information.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the XML driver.

Driver Requirements

You must have the Microsoft XML parser, `msxml6.dll`, installed. If you need to download the file, go to the site:

http://www.microsoft.com

On the Microsoft site, search on "msxml6.dll". Select the link for downloading the parser.

Supported Tabular Formats for XML Documents

The three main XML tabular-formats that the XML driver can access are described in the following table. In some instances, you may need to define hints to help the XML driver read the tabular-format of an XML document correctly. See Configure Location Dialog Box Descriptions on page 304.

Table 31: Common Tabular Formats for XML Documents

Format	Description
ADO 2.5 persisted files	These files are identified by a unique schema namespace URL. Although ADO uses the same data types defined by XML-Data, the data types use extensions, such as adding a maximum column width for string columns. ADO 2.5 persisted files are identified by the following unique XML element: `<xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882" xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882" xmlns:rs="urn:schemas-microsoft-com:rowset" xmlns:z="#RowsetSchema">`
DataDirect Format	This XML format conforms to the W3C recommendation for XML schema, Working Draft April 07, 2000. These files are identified by the following unique XML element (schema namespace URL): `<table targetNamespace= "http://www.merant.com/namespaces/datadirect/xmlrecordset" xsi:schemaLocation= "<http://www.merant.com/namespaces/datadirect/xmlrecordset/EMP.xml>" xmlns="http://www.w3.org/1999/XMLSchema" xmlns:xsi= "http://www.w3.org/1999/XMLSchema-instance" xmlns:rs= "http://www.merant.com/namespaces/datadirect/xmlrecordset"><table xmlns="http://www.w3.org/1999/XMLSchema" xmlns:xsi= "http://www.w3.org/1999/XMLSchema-instance" xmlns:rs= "http://www.datadirect-technologies.com/namespaces /datadirect/xmlrecordset">`
Microsoft Data Islands	These islands are identified by the <XML> tag in an HTML document. The Data Island can be embedded in the HTML document. Data Islands can include the following Schema definition and namespace: `<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes">`

Hierarchical-Formatted XML Document Support

The XML driver can be configured so that it supports hierarchical-formatted documents. In this case, the driver assumes that the document that it is accessing can contain more than one table. The driver scans the document to locate all tables; the available tables are visible through a SQLTables operation. Then, the driver does a second scan to gather each table's column information and to determine a data type for each column.

The following is an example of a hierarchical document:

```
<?xml version="1.0"?>
  <purchaseOrder orderDate="1999-10-20">
```

```
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
     <name>Robert Smith</name>
     <street>8 Oak Avenue</street>
     <city>Old Town</city>
     <state>PA</state>
     <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
     <item partNum="872-AA">
        <productName>Lawnmower</productName>
        <quantity>1</quantity>
        <USPrice>148.95</USPrice>
        <comment>Confirm this is electric</comment>
     </item>
     <item partNum="926-AA">
        <productName>Baby Monitor</productName>
        <quantity>1</quantity>
        <USPrice>39.98</USPrice>
        <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```

First, the XML driver returns two tables: "purchaseOrder" and "items." Two tables are returned because two items are found for a single purchase order. The XML driver found commonality of child elements.

Second, the XML driver determines which columns are in a specific table. An `_ID` column, which is essentially a primary key, is automatically generated for each table. If a table is determined to be a child of another table, then it is given a second generated column. The name of this column is prefixed with the parent table's name and ends with `_ID`, for example, `_purchaseOrder_ID`.

Consider the previous example document. The items table will receive two generated columns, `_ID` and `_purchaseOrder_ID`, which are assigned an integer data type. The purchaseOrder table receives only the `_ID` column, because it does not have a parent table.

The tables returned from the example file include the following columns:

Table	Columns

items	_ID _purchaseOrder_ID partNum productName	quantity USPrice comment Date
purchaseOrder	_ID orderDate shipTo_country shipTo_name shipTo_street shipTo_city shipTo_state shipTo_zip	billTo_country billTo_name billTo_street billTo_city billTo_state billTo_zip comment

Column Data Types

The XML driver determines the column data types by inspecting the column values. The data type determination limits its data types to a subset of the DataDirect Format data types, as listed in the following table. For a complete list of DataDirect Format data types, see Supported Tabular Formats for XML Documents on page 285.

Data Type	Sample Values
wvchar	"Foo", "best320"
varbinary	"27AB2F9C"
int	"34","-7000"
unsignedint	"0", "123456789"
long	"-12345678012345"
unsignedlong	"123456789012345"
boolean	"true", "false"
date	1963-12-19
time	10:09:58
timeinstant	1963-12-19T10:09:58
decimal	1245.678

Defining Locations

When configuring an XML data source, you must define the location of the XML or HTML documents that the driver will access. The locations can be either from a local file system or from a Web server.

The following table describes the types of locations:

Folder	Implies that each XML file is a single table. When defining a Folder location, you specify only a directory as the location (not a directory and a file name), for example, C:\xmlsample.
XML Document	Implies that the full path to the XML document, including the XML file name, is the location. Using this type of location, each document can have one or more tables and can be a hierarchical-formatted XML document. When defining an XML Document location, you specify a path and an XML file name as the location, for example: C:\xmlsample\file.xmlYou can also specify a web service through a URL, for example: http://xxx.company.com/search=XML&mode=books
HTML Document	Implies the use of an HTML document with embedded XML Data Islands. Using this type of location, each document can have one or more tables. When defining an HTML Document location, you specify a path and an HTML file name, for example, C:\htmlsample\file.html, as the location.

Specifying Table Names in SQL Statements

When defining locations, you specify a name for the location along with a directory, or path and file name. For example, suppose you define two locations for a data source, a Folder location and an XML Document location. The Folder location is on a local filing system and the XML Document location is on a web server with a URL prefix of http://www.acme.com/xmldata.

For example:

The Folder location:

c:\xmldata\xmlsample as LOC1

The XML Document location: http://www.acme.com/xmldata/doc.xml as LOC2

For complete information about how to configure locations in an XML data source, see Data Source Configuration through a GUI (XML) on page 290.

If you are connected to this data source and the data source had the "Show Manufactured Schemas" option set as the Schema Mode (see the Schema Mode option under Data Source Configuration through a GUI (XML) on page 290) and then you performed an unqualified SQLTables operation, you would get the following results.

Schema name	Table name
LOC1#	FILE1
LOC1#	FILE2

LOC2#	TABLE1
LOC2#	TABLE2

Location names are fabricated into the schema name by adding a # symbol to the end of the location name.

Note: If you had the "Show Virtual Schemas" option set, the above table would have "XML" listed in the Schema name column.

To fully qualify a table name in a SQL statement, you could use the following:

LOC1#.FILE1

or

XML.FILE1

LOC2#.TABLE2

or

XML.TABLE2

This design gives you a simpler table name qualifier. This is an important advantage given the complexity of URL names, and the requirement to double quote them in SQL statements. For example, the following query uses a fully qualified table name for an XML Document location:

```
SELECT * FROM "http://www.acme.com/xmldata/doc.xml#TABLE2" WHERE productName='lawnmower'
```

Compare that to the same query using a location name:

```
SELECT * FROM LOC2#.TABLE2 WHERE productName='lawnmower'
```

Another example demonstrating the Folder location is as follows:

```
SELECT * FROM "c:\xmldata\xmlsample\FILE1.XML" WHERE productName='lawnmower'
```

Compare that to the same query using a location name:

```
SELECT * FROM LOC1#.FILE1 WHERE productName='lawnmower'
```

Configuring and Connecting to Data Sources (XML)

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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. See Data Source Configuration through a GUI (XML) on page 290 for details.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 296 and Connection Option Descriptions on page 297 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration through a GUI (XML)

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 XML data source:

1. Start the ODBC Administrator by selecting its icon from the DataDirect Connect program group; then, select a tab:

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

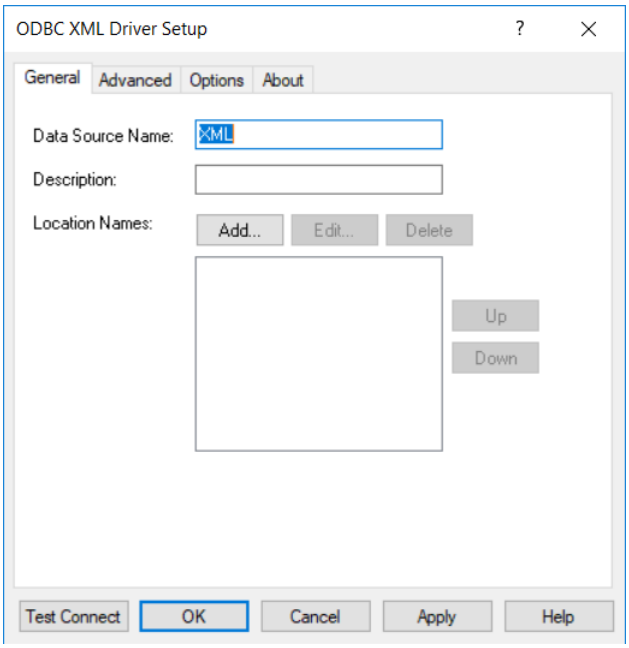
   If you are configuring a new user data source, click **Add** on the User DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

   - **System DSN**: To configure a new system data source, click **Add** on the System DSN tab 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 name on the File DSN tab and click **Configure** to display the driver Setup dialog box.

   If you are configuring a new file data source, click **Add** on the File DSN tab to display a list of installed drivers. Select the driver and click **Next**. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

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

Figure 26: Driver Setup: General tab



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

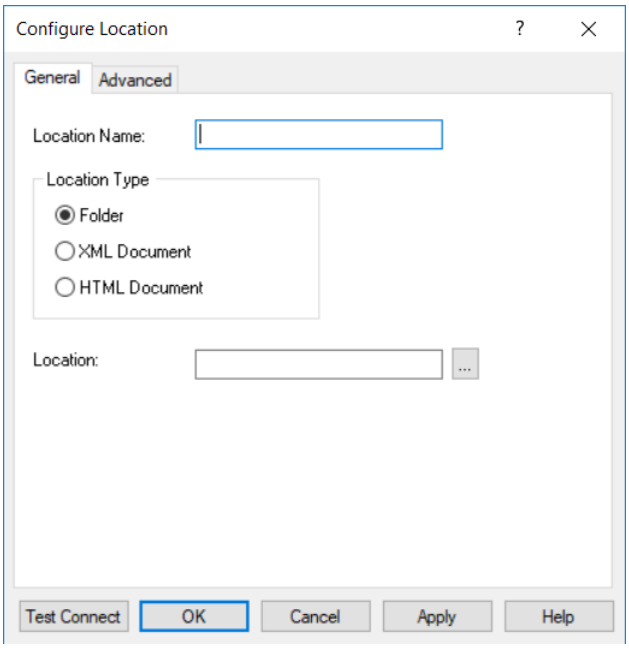
2. On the General tab, provide the following information; then, click **Apply**.

Connection Options: General	Default
Data Source Name on page 297	None
Description on page 298	None
Location Names on page 299	None

3. If you want to edit or delete a location name, or change its position in the list, select it; then, click **Edit**, **Delete**, **Up**, or **Down** as appropriate.

4. If you want to define a location, click **Add**. The Configure Location dialog box appears.

Figure 27: Configure Location: General tab



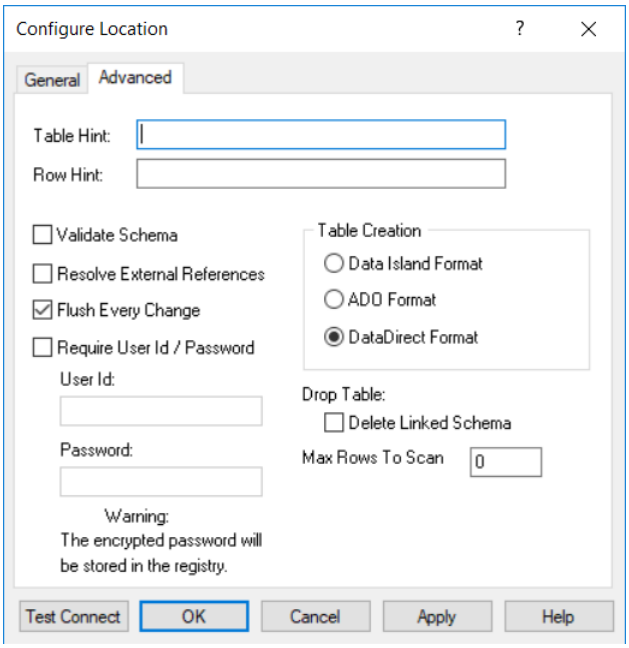
5. On the General tab of the Configure Location dialog box, provide the following required information; then, click **Apply**.

Configure Location Options: General	Default
Location Names on page 299	None
Location Type on page 307	None
Location on page 306	localhost

Location: Either type the full path to the location you are defining or click the select button: ... to select a path.

6. Optionally, click the **Advanced** tab of the Configure Location dialog box to specify additional information about the location you are defining.

Figure 28: Configure Location: Advanced tab



On this tab, provide any of the following optional information; then, click **Apply**.

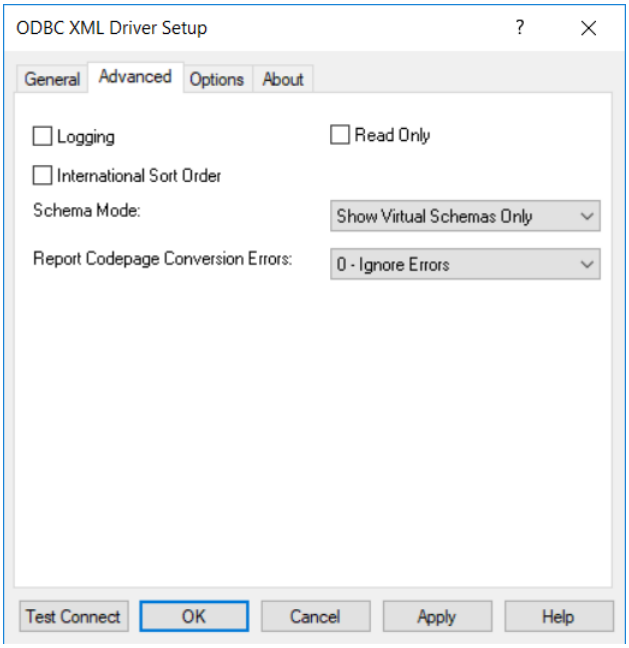
Configure Location Options: Advanced	Default
Table Hint on page 311	None
Row Hint on page 309	None
Validate Schema on page 312	Disabled
Resolve External References on page 309	Disabled
Flush Every Change on page 305	Enabled
Require User ID/Password on page 308	Disabled
User ID on page 311	None
Password on page 308	None
Table Creation on page 310	DataDirect Format
Delete Linked Schema on page 304	Disabled
Max Rows to Scan on page 307	0

7. You can click **Test Connect** to attempt to connect to the location.
   - If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.

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

8. Click **OK** to return to the ODBC XML driver Setup dialog box or **Cancel**. If you click **OK**, the values you have specified become the defaults for this location.
9. Optionally, click the **Advanced** tab of the ODBC XML driver Setup dialog box to specify data source settings.

Figure 29: Driver Setup: Advanced tab

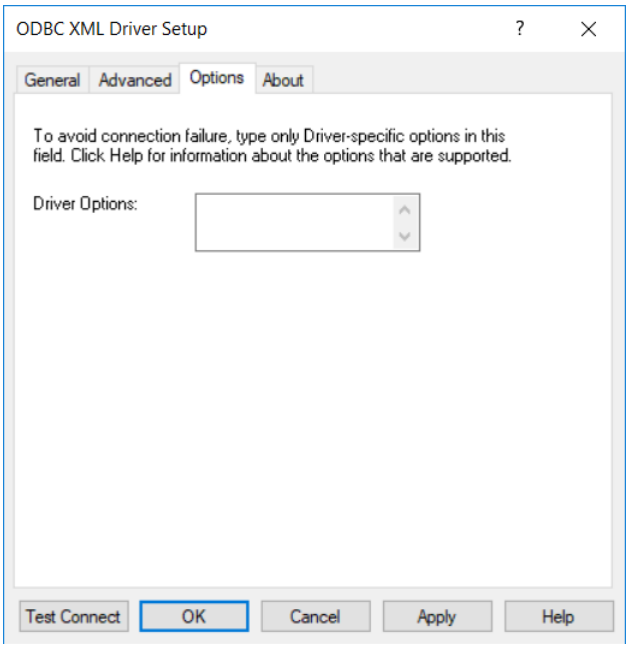


On this tab, provide any of the following optional information; then, click **Apply**.

Connection Options: Advanced	Default
Logging on page 300	Disabled
International Sort Order on page 299	Disabled
Schema Mode on page 302	Show Virtual Schemas Only
Report Codepage Conversion Errors on page 301	0 - Ignore Errors
Read Only on page 300	Disabled

10. Optionally, click the **Options** tab to specify data source connection values.

Figure 30: Driver Setup: Options tab



Driver Options: Type configuration options specific to the XML driver.

Connection Options: Options	Default
Driver Options on page 298	Disabled

Warning: The properties you set in the Options tab override other properties for this session only and can adversely affect the operation of the XML driver. Use only authorized entries. For information about authorized entries for the Options tab, contact Progress DataDirect customer support.

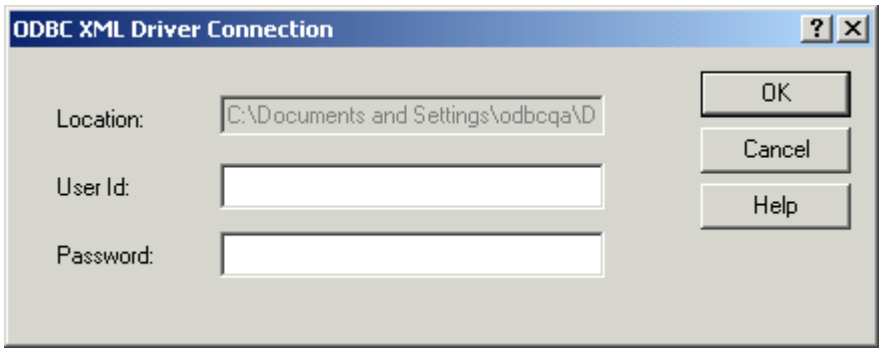
11. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see Using a Logon Dialog Box (XML) on page 295 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

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

12. Click **OK** or **Cancel**. If 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.

Using a Logon Dialog Box (XML)

Some ODBC applications display a Logon dialog box when you are connecting to a data source. For XML, the dialog box is as follows:



This dialog box appears for each password-protected location that you have defined for the data source.

In this dialog box, provide the following information:

1. Type your user ID and password in the appropriate fields for the Location that appears in the Location field.
2. Click **OK** to connect to the data source.

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 on page 297 and Configure Location Dialog Box Descriptions on page 304 give the names and descriptions of the attributes, as well as the initial default value when the driver is first installed.

An example of a DSN connection string with overriding attribute values for XML is:

```
DSN=XML FILES;LOC1.Create Type=ADO25;Logging=1
```

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

```
FILEDSN=XML.dsn;LOC1.Create Type=ADO25;Logging=1
```

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

```
DRIVER={DataDirect 7.1 XML};LOC1={DataDirect Closed XML ADO Provider}
```

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute 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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Note: XML driver connection string attributes do not use short name equivalents.

The following table lists the connection string attributes associated with General, Advanced, and Options tabs of the XML driver Setup dialog box. The descriptions themselves are listed below the table.

Table 32: Attributes on the XML Driver Setup Dialog Box

Attribute	Default
DataSourceName	None
Description	None
InternationalSort	Disabled
Location	None
Logging	Disabled
ReadOnly	Disabled
ReportCodepageConversionErrors	0 (Ignore Errors)
ShowManufacturedSchemas	0 (Disabled)
ShowVirtualSchemas	1 (Enabled)

Data Source Name

Attribute

DataSourceName

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

Driver Setup: General tab

Description

Attribute

Description

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

Driver Setup: General tab

Driver Options

Attribute

n/a

Purpose

Type configuration options specific to the XML driver.

Valid Values

WARNING: The properties you set in the Options tab override other properties for this session only and can adversely affect the operation of the XML driver. Use only authorized entries. For information about authorized entries for the Options tab, contact Progress DataDirect customer support.

Default

None

GUI Tab

Driver Setup: Options tab

International Sort Order

Attribute

InternationalSort

Purpose

Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default

0 (Disabled)

GUI Tab

Driver Setup: Advanced tab

Location Names

Attribute

Location

Purpose

A display of all existing location names defined for the data source you are configuring.

Valid Values

string

where:

`string`

is the name of a location.

The location names listed in the text box are used for connections according to the order that they are displayed. If you want to change the order or precedence, use the Up and Down buttons.

Default

None

GUI Tab

Driver Setup: General tab

Logging

Attribute

Logging

Purpose

Creates a log file that logs the SQL execution plan. A value of 0 means no logging is performed.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), a log file is created in the current directory. The default log file name is \Integrator.txt.

If set to 0 (Disabled), no logging is performed.

Default

0 (Disabled)

GUI Tab

Driver Setup: Advanced tab

Read Only

Attribute

ReadOnly

Purpose

Controls whether the driver opens files for Read-Write access or Read-Only access

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver opens XML files for Read-Only access. In this case, the XML driver opens XML files with a Shared Read lock. This allows other connections and applications to read the same XML file that the XML driver has open; however, they cannot write to the XML file.

If set to 0 (Disabled), the XML driver opens XML files for Read-Write access. Opening an XML file for Read-Write access places an exclusive lock on the file. No other connections or applications can open the XML file while the driver has the file open.

Default

0 (Disabled)

GUI Tab

Driver Setup: Advanced tab

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors

Purpose

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

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

Valid Values

0 | 1 | 2

Behavior

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

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

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

Default

0 (Ignore Errors)

GUI Tab

Driver Setup: Advanced tab

Schema Mode

Attribute

n/a

Purpose

Specifies whether to show virtual schemas, manufactured schemas, or both.

Valid Values

Choose one of the following options:

- Show Virtual Schemas Only. This option returns "XML" in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. See Show Virtual Schemas on page 303.
- Show Manufactured Schemas Only. This option returns the manufactured schema names in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. See Show Manufactured Schemas on page 302.
- Show Both Virtual and Manufactured Schemas. This option returns both virtual and manufactured schema names when a SQLTables or SQLColumns operation is performed when connected to a data source.

Default

Show Virtual Schemas Only

GUI Tab

Driver Setup: Advanced tab

Show Manufactured Schemas

Attribute

ShowManufacturedSchemas

Purpose

Returns the manufactured schema names in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. The Location names you define for a data source are manufactured into a schema name by adding a # symbol after the Location names. For example:

Schema Name	Table Name
LOC1#	TAB1A
LOC1#	TAB1B
LOC2#	TAB2A
LOC2#	TAB2B

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), manufactured schema names are returned.

If set to 0 (Disabled), manufactured schema names are not returned.

To return both manufactured and virtual schema names, set this option to 1 (Enabled) and the Show Virtual Schemas option to 1 (Enabled).

Default

0 (Disabled)

GUI Tab

Driver Setup: Advanced tab

Show Virtual Schemas

Attribute

ShowVirtualSchemas

Purpose

Returns "XML" in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. For example:

Schema Name	Table Name
XML	TAB1A
XML	TAB1B
XML	TAB2A
XML	TAB2B

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), virtual schema names are returned.

If set to 0 (Disabled), virtual schema names are not returned.

To return both virtual and manufactured schema names, set this option to 1 (Enabled) and the Show Manufactured Schemas option to 1 (Enabled).

Default

1 (Enabled)

GUI Tab

Advanced tab

Configure Location Dialog Box Descriptions

The following table lists the connection string attributes associated with General and Advanced tabs of the XML driver Configure Location dialog box. The descriptions themselves are listed below the table. See Defining Locations on page 288 for an explanation of locations.

Note: XML driver connection string attributes do not use short name equivalents.

The names of all connection options in this section are preceded by *location_name*, where *location_name* represents the name of a specific location that you have defined, for example, LOC1. See the description of the Location Names on page 299 option for details.

Table 33: XML Configure Location Attribute Names

Attribute	Default
location_name	None
location_name.Catalog Type Hint	Folder
location_name.Create Type	DataDirect
location_name.Delete Schema	0 (Disabled)
location_name.Flush Every Change	1 (Enabled)
location_name.Initial Catalog	None
location_name.Password	None
location_name.Require Passwd	0 (Disabled)
location_name.Resolve External	0 (Disabled)
location_name.Row Hint	None
location_name.Scan Rows	0
location_name.Table Hint	None
location_name.User ID	None
location_name.Validate Schema	0 (Disabled)

Delete Linked Schema

Attribute

location_name.Delete Schema

Purpose

Specifies whether an externally-linked schema file is deleted when a table is deleted. This option is valid only for Folder location types. The XML document for the table contains a link to this external schema file. By default, this check box is not selected.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the externally-linked schema file is deleted when the table is deleted. If multiple XML documents are linked to the same schema file, the schema file is not deleted when a table is deleted.

If set to 0 (Disabled), the externally-linked schema file is not deleted when the table is deleted.

Default

0 (Disabled)

GUI Tab

Configure Location: Advanced tab

Flush Every Change

Attribute

location_name.Flush Every Change

Purpose

Writes the data document to disk after every insert, update, or delete operation. This option is valid only for Folder location types.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver writes the data document to disk after every change.

If set to 0 (Disabled), the driver does not write the data document to disk after every change. Disabling this option can improve performance.

Default

1 (Enabled)

GUI Tab

Configure Location: Advanced tab

Location

Attribute

location_name.Initial Catalog

Purpose

The full path name to the location you are defining.

Valid Values

location_directory

where location_directory is the full path name of the directory in which the data files are stored. For example:

```
LOC1.Initial Catalog=C:\Documents\filesml
```

Default

None

GUI Tab

Configure Location: General tab

Location Name

Attribute

location_name

Purpose

A unique name for the location you are defining, for example, LOC1.

Valid Values

location_name={DataDirect Closed XML ADO Provider}

where *location_name* is the unique name of the location you are defining. For example, if you choose the location name LOC1, then:

```
LOC1={DataDirect Closed XML ADO Provider}
```

Default

None

GUI Tab

Configure Location: General tab

Location Type

Attribute

location_name.Catalog Type Hint

Purpose

Specifies the type of location you are defining for the connection.

Valid Values

Folder | XML Document | HTML Document

For example:

```
LOC1.Catalog Type Hint=XML Document
```

Default

Folder

GUI Tab

Configure Location: General tab

See also

See Defining Locations on page 288 for the definition of each type.

Max Rows to Scan

Attribute

location_name.Scan Rows

Purpose

An integer that represents the maximum number of rows to scan when the XML driver is determining the data type of each column. This option is valid only for XML Document location types.

Valid Values

0 | x

where:

x

is the number of rows to scan.

Behavior

If set to x, the driver scans a maximum of x rows in the table. During the scan, the driver inspects each column value in the row of a table and adjusts the data type determination for each column based on the corresponding value. The more sample column values it encounters, the more accurate the determination.

If set to 0, the driver scans all rows in the table. Disabling this option can improve performance because limiting the number of rows can reduce the amount of time it takes to determine the column information on very large documents. Because less information is available, however, the determination of the data types can be incorrect.

Default

0

GUI Tab

Configure Location: Advanced tab

Password

Attribute

location_name.Password

Purpose

The password used to establish a connection to the location specified by *location_name*. A password is required only if the location to which you are connecting is password-protected.

This option is not available unless the Require User ID/Password option is enabled.

Valid Values

pwd

where:

pwd

is a valid password.

Warning: The encrypted password is stored in the Windows Registry.

Default

None

GUI Tab

Configure Location: Advanced tab

Require User ID/Password

Attribute

location_name.Require Passwd

Purpose

Specifies whether a User ID and password are required to establish a connection to the location you are defining.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), a User ID and password are required to establish a connection to the location. You must enable this option if the location you are defining is password-protected; otherwise, the connection will fail. Enabling this option causes a Logon dialog box to appear when connecting with the driver.

If set to 0 (Disabled), no user ID and password are required to establish a connection to the location.

Default

0 (Disabled)

GUI Tab

Configure Location: Advanced tab

Resolve External References

Attribute

location_name.Resolve External

Purpose

Determines whether external references such as DTDs, Schemas, Entities, and Notations are resolved for the XML documents contained within the location specified by *location_name*.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the documents are not processed if the XML parser cannot locate the external references.

If set to 0 (Disabled), the document is processed, even if the XML parser cannot locate the external references.

Default

0 (Disabled)

GUI Tab

Configure Location: Advanced tab

Row Hint

Attribute

location_name.Row Hint

Purpose

A string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the nodes that make up the rows in the rowset of a tabular-formatted XML document contained within the location specified by *location_name*. See Using Hints for Tabular-Formatted XML Documents on page 312 for details.

This option is valid only for Folder and HTML Document location types.

Valid Values

row_hint

where:

`row_hint`

is an XSL pattern.

Default

None

GUI Tab

Configure Location: Advanced tab

Table Creation

Attribute

location_name.Create Type

Purpose

Determines the style of XML that is generated when a new table is created. This option is valid only for Folder location types.

Valid Values

IE5DataIsland | ADO25 | DataDirect

- Data Island Format (IE5DataIsland): New tables are created with the Internet Explorer 5 Data Island XML style.
- ADO Format (ADO25): New tables are created with the ADO 2.5 XML style.
- DataDirect Format (DataDirect): New tables are created with the DataDirect format. This format conforms to the W3C recommendation for XML schema, working draft April 07, 2000.

Default

DataDirect

GUI Tab

Configure Location: Advanced tab

See also

See Supported Tabular Formats for XML Documents on page 285 for a description of each of these formats.

Table Hint

Attribute

location_name.Table Hint

Purpose

A string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the table or rowset nodes in a tabular-formatted XML document contained within the location specified by *location_name*. See Using Hints for Tabular-Formatted XML Documents on page 312 for details.

This option is valid only for Folder and HTML Document location types.

Valid Values

table_hint

where:

`table_hint`

is an XSL pattern.

Default

None

GUI Tab

Configure Location: Advanced tab

User ID

Attribute

location_name.User ID

Purpose

The User ID (user name) used to establish a connection to the location specified by *location_name*. A password is required only if the location to which you are connecting is password-protected.

This option is not available unless the Require User ID/Password option is enabled.

Valid Values

userid

where:

`userid`

is a valid user name.

Default

None

GUI Tab

Configure Location: Advanced tab

Validate Schema

Attribute

location_name.Validate Schema

Purpose

Determines whether the XML documents contained within the location specified by *location_name* are validated against their schema.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the XML documents are validated against their schema. This allows a well-formed XML document to be processed, even if the document is not valid.

If set to 0 (Disabled), the XML documents are not validated against their schema.

Default

0 (Disabled)

GUI Tab

Configure Location: Advanced tab

Using Hints for Tabular-Formatted XML Documents

The XML driver supports table and row hints. You can specify a table hint, a row hint, or both, when configuring an XML data source or using a connection string.

Table hints should be specified so that they resolve to a single node. If a table hint resolves to a set of nodes, the first node in the set is used as the table node. The context of the table hint is always the root node of the XML document.

Row hints define the "row" element and specify whether the rowset is element-based or attribute-based. If a table hint is supplied, the context of the row node is the node to which the table hint resolves; otherwise, the context is the root node of the XML document. The column mode identifier specifies whether the columns of a row are child nodes or attributes of the row node.

When working with hints, keep in mind that the XML driver assumes that the row nodes are the immediate children of the table node.

- If only a table hint is specified, the row nodes are the children of the node to which the hint resolves. It is assumed that all of the child nodes have the same name.
- If only a row hint is specified, the table node is the parent of the node to which the hint resolves. If the row hint resolves to a set of nodes, the nodes in that set must all have the same parent.

- If both a table hint and a row hint are specified, the row hint is taken to be relative to the node to which the table hint resolves.

The column mode identifier has the format:

```
\column mode
```

where mode can be one of the following options:

- child: The columns are child nodes of the row node.
- attr: The columns are attributes of the row node.

In the following examples, the columns are the children of the row nodes.

Example 1

Table Hint:

Row Hint: //Item

The row nodes are the nodes named Item. The table node is the parent of the row nodes. Use this form only when all of the Item nodes reside under one parent.

If some Item nodes have different parents, use a table hint or a more specific row hint to select the set of Item nodes.

Example 2

Table Hint:

Row Hint: /Bookstore/Books/Item

The row nodes are the nodes named Item. The table node is Books, which is a child of the Bookstore node.

Example 3

Table Hint: /Bookstore/Books

Row Hint:

The table node is Books, which is a child of the Bookstore node. The row nodes are the children of the Books node. It is assumed that all of the child nodes under the Books nodes have the same name. If the child nodes do not all have the same name, the name of the first child node encountered is used as the row node name. In that case, it would be better to specify both a table and row hint.

Example 4

Table Hint: /Bookstore [@location = "Raleigh"]/Books

Row Hint: ./Item

The table node is Books, which is a child of the Bookstore node. Bookstore has a "location" attribute with the value Raleigh. The row nodes are the Item nodes that are children of the Books node.

Column Mode Identifier

The following examples illustrate the use of the optional column mode identifier.

Example 5

Table Hint:

Row Hint: //Item \column attr

The row nodes are named Item. The table node is the parent of the row nodes. The columns are attributes of the row node.

Example 6

Table Hint:

Row Hint: //Item \column child

The row nodes are the nodes named Item. The table node is the parent of the row nodes. The columns are attributes of the row node.

Data Types

This section provides three tables that show how the data types for each supported tabular-formatted XML document map to the standard ODBC data types.

Table 34: Data Islands Data Types

Data Islands	Internal XML Name	ODBC
binhex	bin.hex	SQL_LONGVARBINARY
boolean	boolean	SQL_BIT
currency	fixed.14.4	SQL_DECIMAL
date	date	SQL_TYPE_DATE
dateTime	dateTime	SQL_TYPE_TIMESTAMP
float	float	SQL_DOUBLE
i1	i1	SQL_TINYINT SIGNED
i2	i2	SQL_SMALLINT SIGNED
i4	i4	SQL_INTEGER SIGNED
int	int	SQL_INTEGER SIGNED
number	number	SQL_DOUBLE
r4	r4	SQL_REAL
r8	r8	SQL_DOUBLE
singleChar	singleChar	SQL_SMALLINT
string	string	SQL_WLONGVARCHAR
time	time	SQL_TYPE_TIME

Data Islands	Internal XML Name	ODBC
ui1	ui1	SQL_TINYINT UNSIGNED
ui2	ui2	SQL_SMALLINT UNSIGNED
ui4	ui4	SQL_INTEGER UNSIGNED

Table 35: ADO 2.5 Persisted Files Data Types

ADO 2.5 Persisted Files	Internal XML Name	ODBC
binhex	bin.hex	SQL_LONGVARBINARY
boolean	boolean	SQL_BIT
currency	fixed.14.4	SQL_DECIMAL
date	date	SQL_TYPE_DATE
dateTime	dateTime	SQL_TYPE_TIMESTAMP
float	float	SQL_DOUBLE
i1	i1	SQL_TINYINT SIGNED
i2	i2	SQL_SMALLINT SIGNED
i4	i4	SQL_INTEGER SIGNED
i8	i8	SQL_BIGINT SIGNED
int	int	SQL_INTEGER UNSIGNED
number	number	SQL_DOUBLE
r4	r4	SQL_REAL
r8	r8	SQL_DOUBLE
singleChar	singleChar	SQL_SMALLINT SIGNED
time	time	SQL_TYPE_TIME
ui1	ui1	SQL_TINYINT UNSIGNED
ui2	ui2	SQL_SMALLINT UNSIGNED
ui4	ui4	SQL_INTEGER UNSIGNED
ui8	ui8	SQL_BIGINT UNSIGNED
wchar	string	SQL_CHAR

ADO 2.5 Persisted Files	Internal XML Name	ODBC
wchar	string	SQL_WCHAR
wlvarchar	string	SQL_WLONGVARBINARY
wvarchar	string	SQL_WVARCHAR

Table 36: DataDirect Format Data Types

DataDirect	Internal XML Name	ODBC
binary	binary	SQL_BINARY
boolean	boolean	SQL_BIT
byte	byte	SQL_TINYINT SIGNED
date	date	SQL_TYPE_DATE
decimal	decimal	SQL_NUMERIC
double	double	SQL_DOUBLE
float	float	SQL_REAL
int	int	SQL_INTEGER UNSIGNED
long	long	SQL_BIGINT SIGNED
lvarbinary	binary	SQL_LONGVARBINARY
short	short	SQL_SMALLINT SIGNED
time	time	SQL_TYPE_TIME
timeInstant	timeInstant	SQL_TYPE_TIMESTAMP
unsignedByte	unsignedByte	SQL_TINYINT UNSIGNED
unsignedInt	unsignedInt	SQL_INTEGER UNSIGNED
unsignedLong	unsignedLong	SQL_BIGINT UNSIGNED
unsignedShort	unsignedShort	SQL_SMALLINT UNSIGNED
varbinary	binary	SQL_VARBINARY
wchar	string	SQL_CHAR
wchar	string	SQL_WCHAR

DataDirect	Internal XML Name	ODBC
wlvarchar	string	SQL_WLONGVARBINARY
wvarchar	string	SQL_WVARCHAR

See Retrieving Data Type Information on page 47 for information about retrieving data types.

Unicode Support

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

See UTF-16 Applications on UNIX and Linux on page 101 for related details.

Also, refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for a more detailed explanation of Unicode.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See Persisting a Result Set as an XML Data File on page 48 for details about implementation.

ODBC Conformance Level

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

In addition, the driver supports SQLSetPos.

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

Number of Connections and Statements Supported

There is no limit to the number of connections and statements supported.

SQL Support

This section provides information about the SQL statements that the XML driver processes, and about SQL standards and conventions that the driver supports:

- SQL Statements on page 318
- Extensions to SQL Standards on page 318
- Grammar Token Definitions on page 318

SQL Statements

The SQL Engine included with the XML driver supports the following SQL statements:

- Select
- Create and Drop Table
- Insert
- Update
- Delete

Note: See the table at the beginning of this chapter for the SQL statements that the XML driver supports for the different types of supported file formats.

Extensions to SQL Standards

The XML driver uses SQL grammar that is compliant with entry level ANSI SQL-92. The following table summarizes significant extensions to the grammar.

Table 37: SQL Extensions

Entry Level ANSI SQL-92 Extension	Relevant Standard or Convention
Aliasing table references	Intermediate level ANSI SQL-92
ANSI date, time, and timestamp literals	Intermediate level ANSI SQL-92
Dynamic parameter specification	Full level ANSI SQL-92
GUID literals	COM
Hex string literals	Full level ANSI SQL-92
Left Outer Joins	Intermediate level ANSI SQL-92
ODBC escape support	ODBC 3.0
Scalar functions	ODBC 3.0

Grammar Token Definitions

The tokens used in the XML driver SQL grammar are defined in the following sections:

- Regular Identifiers on page 319
- Delimited Identifiers on page 319
- Integer Numbers on page 319
- Real Numbers on page 319
- Character String Literals on page 320

Regular Identifiers

A regular identifier must begin with a letter and may not exceed 128 characters. In addition, all ASCII characters are converted to uppercase.

The following are examples of regular identifiers:

- FOO
- COLUMN_NAME
- SCHEMA#NAME
- Col3 (legal, but converted to COL3)

Delimited Identifiers

Delimited identifiers may not exceed 128 characters. A double quotation character can be embedded within the string by specifying two consecutive double quotation mark characters. A delimited identifier can span multiple lines. The body of a delimited identifier can contain any character except the newline character.

The following examples show delimited identifiers:

- "$ % ^ ($"
- "This is a delimited variable name"

Integer Numbers

Examples of integer numbers are:

- 5
- 1004

Real Numbers

Examples of real numbers are:

- .10
- 12.01
- 10.
- .01e-10
- 12E+10

- 12.01e2
- 12.01e-10
- 12.e-10

Character String Literals

Character string literals are delimited with single quotation mark characters. A single quotation mark character can be embedded within the string by specifying two consecutive single quotation mark characters. A character string literal can span multiple lines.

Examples are:

- '$%^($'
- 'This is a character string literal’

GUID Literals

A GUID uses the following format, where x is a hexadecimal digit:

```
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Hex Literals

Hex literal values are introduced with an uppercase X followed by a single quoted string of hexadecimal characters.

Examples are:

- X’39FA’
- X’B0F00D’

Time and Date Literals

Date, time, and timestamp literals are date, time, and timestamp values surrounded by a standard prefix and suffix. Date literals are specified in a *YYYY-MM-DD* format. Time literals are specified in an *HH:MM:SS* format with an optional fraction component. Timestamp literals are a concatenation of date and time values.

Examples for ODBC and SQL syntax are shown in the following table.

Table 38: Time and Date Literals

Literal Type	ODBC Syntax	ANSI SQL-92 Syntax
Date Literal	`{d '1999-09-19'}`	`date '1999-09-19'`
Time Literal	`{t '11:11:11.225'}`	`time '11:11:11.225'`
Timestamp Literal	`{ts '1999-09-19 11:11:11.225'}`	`timestamp '1999-09-19 11:11:11.225'`
Timestamp Literal	`{ts '1999-09-19'}`	`timestamp '1999-09-19'`

Note: ODBC 1.*x* style ODBC escape sequences such as the following are not supported:
`--(*VENDOR(Microsoft), PRODUCT(ODBC) ...*)--`

SQL Operators and Symbols

Table 39: SQL Operators and Symbols

Symbol	Description	Symbol	Description
':'	Colon	'<'	Less than operator
';'	Semicolon	‘)’	Right parenthesis
'.'	Period	‘=’	Equal operator
','	Comma	‘+’	Plus operator
'<>'	Not equal operator	‘-‘	Minus operator
'<='	Less than or equal operator	‘*’	Multiply operator
'>='	Greater than or equal operator	‘/’	Divide operator
‘>’	Greater than operator	‘?’	Dynamic parameter
‘(‘	Left parenthesis		

Keywords for the XML Driver

A keyword may not be used as a regular identifier. For example, the following statement would generate a syntax error because INDICATOR is a keyword:

```
SELECT INDICATOR FROM T1
```

You can, however, enclose a keyword in double quotation marks to form a delimited identifier. For example, the following statement is valid:

```
SELECT "INDICATOR" FROM T1
```

The following table lists all of the keywords that are reserved for use in SQL statements or designated as potential future reserved words.

Table 40: Reserved Keywords

ABSOLUTE	ACTION	ADD
AFTER	ALIAS	ALL
ALLOCATE	ALTER	AND
ANY	ARE	AS
ASC	ASSERTION	ASYNC
AT	AUTHORIZATION	AVG
BEFORE	BEGIN	BETWEEN
BIT	BIT_LENGTH	BOOLEAN
BOTH	BREADTH	BY
CALL	CASCADE	CASCADED
CASE	CAST	CATALOG
CHAR	CHAR_LENGTH	CHARACTER
CHARACTER_ LENGTH	CHECK	CLOSE
COALESCE	COLLATE	COLLATION
COLUMN	COLUMNS	COMMIT
COMPLETION	CONCAT	CONNECT
CONNECTION	CONSTRAINT	CONSTRAINTS
CONTINUE	CONVERT	CORRESPONDING
COUNT	CREATE	CROSS
CURDATE	CURRENT	CURRENT_DATE
CURRENT_TIME	CURRENT_ TIMESTAMP	CURRENT_USER
CURSOR	CURTIME	CYCLE

DATA	DATE	DAY
DAYOFMONTH	DAYOFWEEK	DEALLOCATE
DEC	DECIMAL	DECLARE
DEFAULT	DEFERRABLE	DEFERRED
DELETE	DEPTH	DESC
DESCRIBE	DESCRIPTOR	DIAGNOSTICS
DICTIONARY	DISCONNECT	DISTINCT
DOMAIN	DOUBLE	DROP
EACH	ELSE	ELSEIF
END	END_EXEC	EQUALS
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FLOOR	FOR	FOREIGN
FOUND	FROM	FULL
GENERAL	GET	GLOBAL
GO	GOTO	GRANT
GROUP	HAVING	HOUR
IDENTIFY	IF	IFNULL
IGNORE	IMMEDIATE	IN
INDEX	INFO	INDICATOR
INITIALLY	INNER	INPUT
INSENSITIVE	INSERT	INT
INTEGER	INTERSECT	INTERVAL
INTO	IS	ISOLATION
JOIN	KEY	LANGUAGE

LAST	LCASE	LEADING
LEAVE	LEFT	LENGTH
LESS	LEVEL	LIKE
LIMIT	LOCAL	LOOP
LOWER	LTRIM	MATCH
MAX	MIN	MINUTE
MOD	MODIFY	MODULE
MONTH	NAMES	NATIONAL
NATURAL	NCHAR	NEW
NEXT	NO	NONE
NOT	NOW	NULL
NULLIF	NUMERIC	OBJECT
OCTET_LENGTH	OF	OFF
OID	OLD	ON
ONLY	OPEN	OPERATION
OPERATORS	OPTION	OR
ORDER	OTHERS	OUTER
OUTPUT	OVERLAPS	PAD
PARAMETERS	PARTIAL	PENDANT
POSITION	POWER	PRECISION
PREORDER	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVATE
PRIVILEGES	PROCEDURE	PROTECTED
PUBLIC	RCASE	READ
REAL	RECURSIVE	REF
REFERENCES	REFERENCING	RELATIVE
REMOVE	REPLACE	RESIGNAL

RESTRICT	RETURN	RETURNS
REVOKE	RIGHT	ROLE
ROLLBACK	ROUND	ROUTINE
ROW	ROWS	RTRIM
SAVEPOINT	SCHEMA	SCROLL
SEARCH	SECOND	SECTION
SELECT	SENSITIVE	SEQUENCE
SESSION	SESSION_USER	SET
SIGNAL	SIMILAR	SIZE
SMALLINT	SOME	SPACE
SQL	SQLCODE	SQLERROR
SQLEXCEPTION	SQLSTATE	SQLWARNING
STRUCTURE	SUBSTRING	SUM
SYSTEM_USER	TABLE	TEMPORARY
TEST	THEN	THERE
TIME	TIMESTAMP	TIMEZONE_ HOUR
TIMEZONE_ MINUTE	TO	TRAILING
TRANSACTION	TRANSLATE	TRANSLATION
TRIGGER	TRIM	TRUE
TYPE	UCASE	UNDER
UNION	UNIQUE	UNKNOWN
UPDATE	UPPER	USAGE
USER	USING	VALUE
VALUES	VARCHAR	VARIABLE
VARYING	VIEW	VIRTUAL
VISIBLE	WAIT	WHEN
WHENEVER	WHERE	WHILE

WITH	WITHOUT	WORK
WRITE	YEAR	ZONE

SQL Comments

ANSI SQL-92 standard comments (--) and C++ standard comments (/*...*/, //) are supported. Comments can be nested.

For example, in the following query columns col2, col3, and col4 are ignored:

```
SELECT col1  /* col1 comment */
/*
    col2,-- col2 comment
    col3,// col3 comment
    col4,/* col4 comment */
*/
FROM t1
```

9

The Connect XE Drivers

This part describes the Progress DataDirect Connect XE drivers. See Drivers Only Available for 32-Bit Platforms on page 239 and Drivers for 32-Bit and 64-Bit Platforms on page 103 for information on additional Connect Series drivers.

For details, see the following topics:

- The Greenplum Wire Protocol Driver
- The Impala Wire Protocol Driver
- The Driver for the Teradata Database

The Greenplum Wire Protocol Driver

The DataDirect Connect XE for ODBC and DataDirect Connect64 XE for ODBC Greenplum Wire Protocol driver (the Greenplum Wire Protocol driver) each support the following database servers:

- Greenplum

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

The Greenplum Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the Windows, UNIX, and Linux environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect XE product for the file name of the Greenplum Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 335 and Greenplum Connection Option Descriptions on page 337 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX/Linux odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration on page 27 for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Greenplum Connection Option Descriptions on page 337 lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (Greenplum)

 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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

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

To configure a Greenplum data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

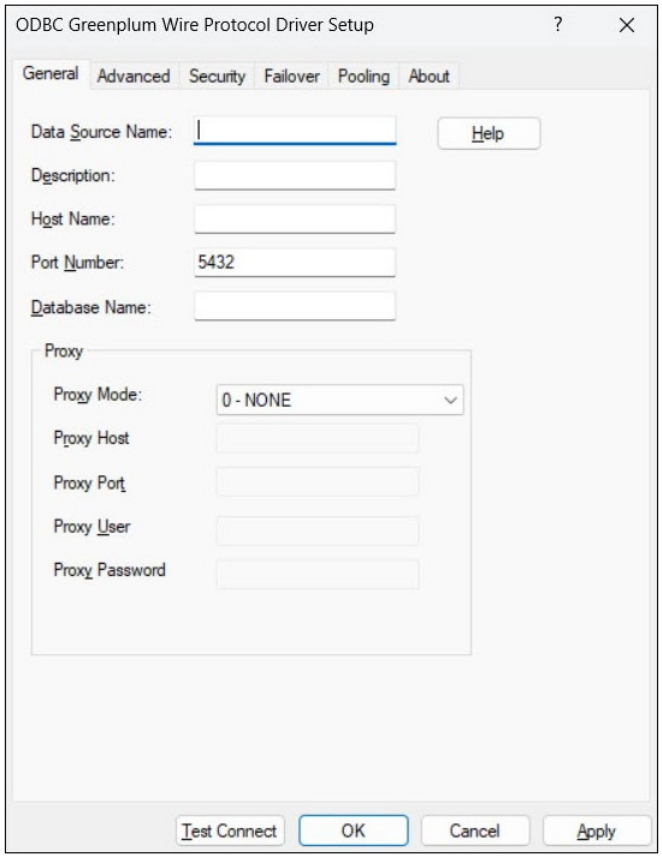
  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **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.

The General tab of the Setup dialog box appears.

Figure 31: General tab



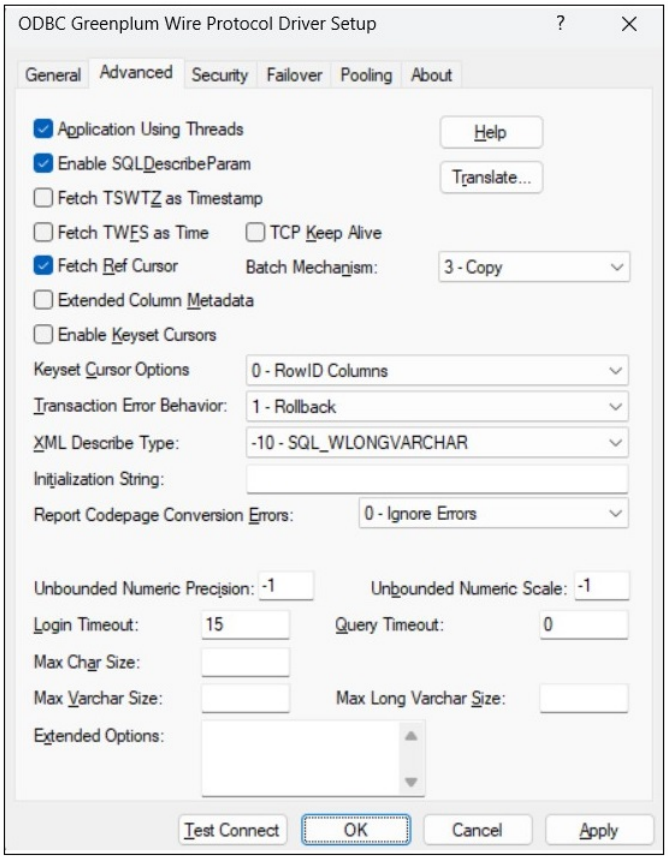
Note: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3. On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 347	None
Description on page 348	None
Host Name on page 357	None
Port Number on page 369	5432
Database Name on page 348	None

4. Optionally, click the **Advanced** tab to specify additional data source settings.

Figure 32: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Using Threads on page 341	Enabled
Enable SQLDescribeParam on page 350	Enabled
Fetch TSWTZ as Timestamp on page 355	Disabled

Connection Options: Advanced	Default
Fetch TWFS as Time on page 355	Disabled
TCP Keep Alive on page 373	Disabled
Fetch RefCursors on page 354	Enabled
Batch Mechanism on page 342	1 - SingleRowInsert
Extended Column MetaData on page 351	Disabled
Enable Keyset Cursors on page 349	Disabled
Keyset Cursor Options on page 360	0 - RowID Columns
Transaction Error Behavior on page 373	1 - Rollback
XML Describe Type on page 378	-10 - SQL_WLONGVARCHAR
Initialization String on page 359	None
Report Codepage Conversion Errors on page 370	0 - Ignore Errors
Unbounded Numeric Precision on page 377	1000
Unbounded Numeric Scale on page 377	6
Login Timeout on page 364	15
Query Timeout on page 370	0
Max Char Size on page 365	None
Max Long Varchar Size on page 365	None
Max Varchar Size on page 367	None
IANAAppCodePage on page 358UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

`Database=Server1;`UndocumentedOption1=*`value`*`[;`UndocumentedOption2=*`value`*`;]`

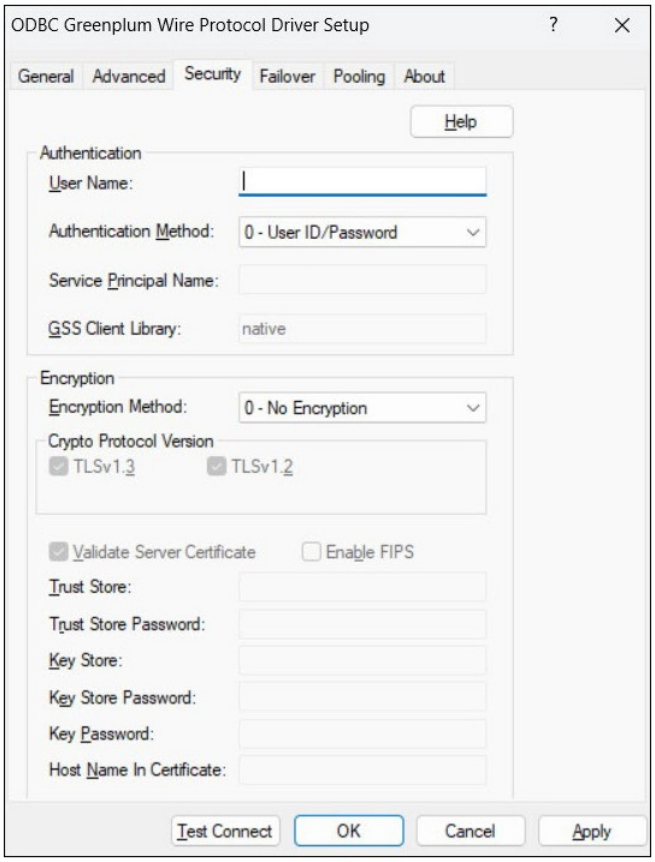
If the Extended Options string contains option values that are also set in the setup dialog or data source, the values of the options specified in the Extended Options string take precedence. However, connection options that are specified on a connection string override any option value specified in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

Note: Do not specify the Extended Options configuration option in a connection string, or the driver will return an error. Instead, applications should specify the individual undocumented connection options in the connection string.

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

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

5. Optionally, click the **Security** tab to specify security data source settings.



See Using Security on page 63 for a general description of authentication and encryption and their configuration requirements.

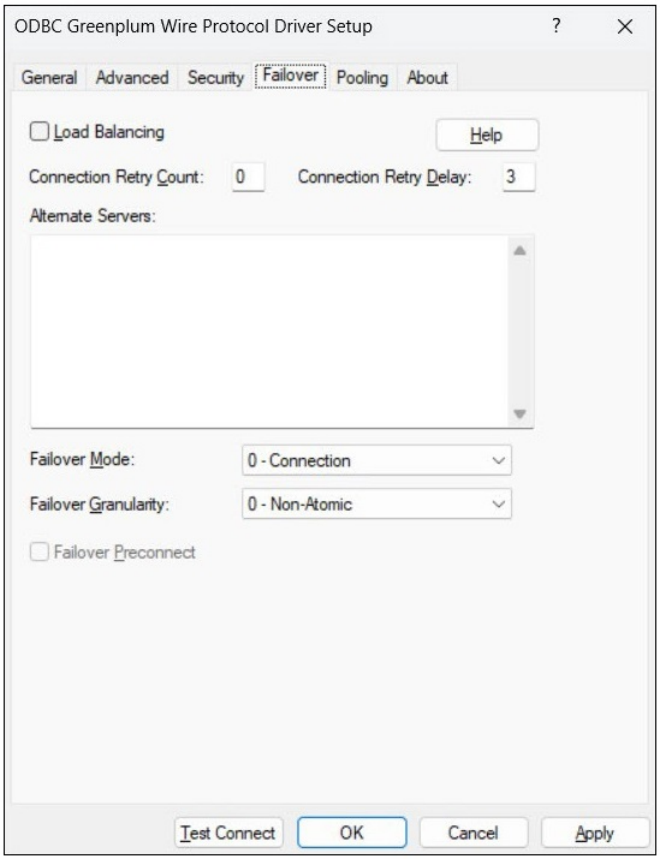
On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Security	Default
User Name on page 378	None
Authentication Method on page 341	0 - No Encryption
Service Principal Name on page 371	None
GSS Client Library on page 356	native

Connection Options: Security	Default
Encryption Method on page 351	0 - No Encryption
Crypto Protocol Version on page 345	TLSv1.2, TLSv1.1, TLSv1, SSLv3
Validate Server Certificate on page 376	1 (Enabled)
Enable FIPS on page 119	Disabled
Truststore on page 374	None
Truststore Password on page 375	None
Keystore on page 361	None
Keystore Password on page 362	None
Key Password on page 361	None
Keystore on page 361	None
Host Name In Certificate on page 358	None

6. Optionally, click the **Failover** tab to specify failover data source settings.

Figure 33: Failover tab



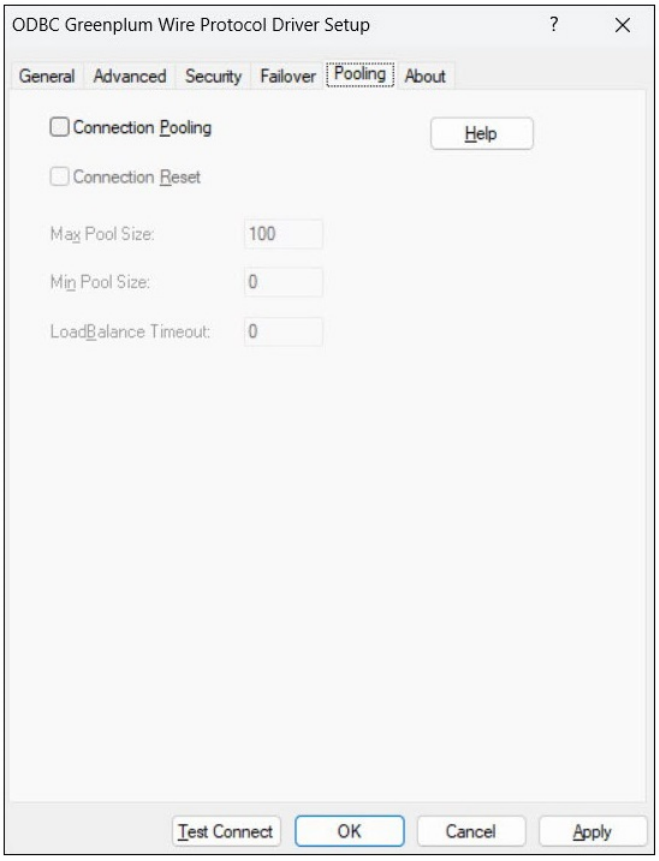
See Using Failover on page 55 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing on page 363	Disabled
Connection Retry Count on page 344	0
Connection Retry Delay on page 345	3
Alternate Servers on page 340	None
Failover Mode on page 353	0 - Connection
Failover Granularity on page 352	0 - Non-Atomic
Failover Preconnect on page 354	Disabled

7. Optionally, click. the **Pooling** tab to specif pooling data source settings.

Figure 34: Pooling tab



See Using DataDirect Connection Pooling on page 73 for a general description of connection pooling.

On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Pooling	Default
Connection Pooling on page 343	Disabled
Connection Reset on page 343	Disabled
Max Pool Size on page 366	100
Min Pool Size on page 366	0
Load Balance Timeout on page 363	0

8. 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 (Greenplum) on page 336 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.
   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

   Note: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

9. Click **OK** or **Cancel**. If 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.

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]...]
```

Greenplum Connection Option Descriptions on page 337 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 Greenplum Wire Protocol is:

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

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

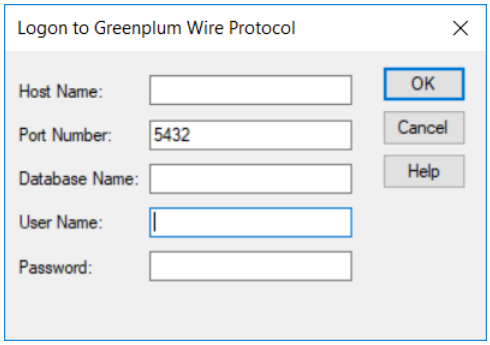
```
FILEDSN=GreenplumWP.dsn;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER={DataDirect 7.1 Greenplum Wire Protocol};
HOST=GreenplumServer;PORT=5432;UID=JOHN;PWD=XYZZY;DB=Gplumdb1
```

Using a Logon Dialog Box (Greenplum)

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



In this dialog box, provide the following information:

1. In the Host Name field, type either the name or the IP address of the server to which you want to connect. The IP address must be in IPv4 format.
2. In the Port Number field, type the number of your Greenplum listener. Check with your database administrator for the correct number.
3. In the Database Name field, type the name of the database to which you want to connect.
4. If required, type your Greenplum user name.
5. If required, type your Greenplum password.
6. Click **OK** to log on to the Greenplum database installed on the server you specified and to update the values in the Registry.

Accessing Greenplum data with Power BI

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

To use the driver to access data with Power BI:

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

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

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

Results:

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

Greenplum 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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the Greenplum Wire Protocol driver.

Table 41: Greenplum Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASRV)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
AuthenticationMethod (AM)	0 (No Encryption)
BatchMechanism (BM)	1 (SingleRowInsert)
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CryptoLibName (CLN)	Empty string
CryptoProtocolVersion (CPV)	TLSv1.2, TLSv1.1, TLSv1, SSLv3
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
EnableFIPS (EF)	Disabled
Enable SQLDescribeParam	1 (Enabled)
EnableKeysetCursors (EKC	0
EncryptionMethod (EM)	0 (No Encryption)
ExtendedColumnMetaData (ECMD)	0 (Disabled)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchRefCursors (FRC)	1 (Enabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
GSSClient (GSSC)	native
HostName (HOST)	None

Attribute (Short Name)	Default
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
KeepAlive (KA)	Disabled
KeysetCursorOptions (KCO)	0 - RowID Columns
KeyPassword (KP)	None
KeystorePassword (KSP)	None
Keystore (KS)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
MaxCharSize (MCS)	None
MaxLongVarcharSize (MLVS)	None
MaxPoolSize (MXPS)	100
MaxVarcharSize (MVS)	None
MinPoolSize (MNPS)	0
OpenSSLConfigFile (OSSLCNF)	*install_dir*\drivers\openssl.cnf (Windows) *install_dir*/lib/openssl.cnf (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*install_dir*\drivers (Windows) *install_dir*/lib (UNIX/Linux)
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
PortNumber (PORT)	5432
QueryTimeout (QT)	0

Attribute (Short Name)	Default
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
ServicePrincipalName (SPN)	None
SSLLibName (SLN)	Empty string
TransactionErrorBehavior (TEB)	1 (Rollback Transaction)
Truststore (TS)	None
TruststorePassword (TSP)	None
UnboundedNumericPrecision (UNP)	1000
UnboundedNumericScale (UNS)	6
ValidateServerCertificate (VSC)	1 (Enabled)
XMLDescribeType (XDT)	-10

Alternate Servers

Attribute

AlternateServers (ASRV)

Purpose

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

Valid Values

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

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

Example

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

```
AlternateServers=(HostName=GreenplumServer:PortNumber=5431:
Database=Pgredb1, HostName=255.201.11.24:PortNumber=5432:Database=Pgredb2)
```

Default

None

GUI Tab

Failover tab

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Default

1 (Enabled)

GUI Tab

Advanced tab

See also

Performance Considerations

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

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

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

Valid Values

0 | 4

Behavior

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

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

Default

0 (No Encryption)

GUI Tab

Security tab

Batch Mechanism

Attribute

BatchMechanism (BM)

Purpose

Determines the mechanism that is used to execute batch operations.

Valid Values

1 | 2 | 3

Behavior

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

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

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

Default

1 (SingleRowInsert)

Notes

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

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 379 for details.

Connection Pooling

Attribute

Pooling (POOL)

Purpose

Specifies whether to use the driver's connection pooling.

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Pooling tab

See also

Performance Considerations on page 379

Connection Reset

Attribute

ConnectionReset (CR)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Pooling tab

See also

Performance Considerations on page 379

Connection Retry Count

Attribute

ConnectionRetryCount (CRC)

Purpose

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

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

Valid Values

0 | x

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

0

GUI Tab

Failover tab

Connection Retry Delay

Attribute

ConnectionRetryDelay (CRD)

Purpose

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

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

Valid Values

0 | x

where:

x

is a positive integer from 1 to 65535.

Behavior

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

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

Default

3

GUI Tab

Failover tab

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.2 | TLSv1.3

Example

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

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

Notes

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

Default

TLSv1.2,TLSv1.1

GUI Tab

Security tab

See also

Encryption Method on page 351

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

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

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

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

Empty string

GUI Tab

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

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

See Advanced tab for details.

See also

SSLLibName on page 372

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

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

Description

Description (n/a)

Purpose

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

Valid Values

string

where:

string

is a description of a data source.

Default

None

GUI Tab

General tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

`0`

Enable Keyset Cursors

Attribute

EnableKeysetCursors (EKC)

Purpose

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

Valid Values

0 | 1

Behavior

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

If set to 0 (Disabled), the driver does not emulate keyset cursors. If an application requests a keyset cursor and this option is set to 0, the driver uses a static cursor and returns a message that a different value was used.

Default

0 (Disabled)

GUI Tab

Advanced tab

Enable SQLDescribeParam

Attribute

EnableDescribeParam (EDP)

Purpose

Determines whether SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

If set to 0 (Disabled), the driver does not support SQLDescribeParam and returns the message: `Driver does not support this function`.

Default

1 (Enabled)

GUI Tab

Advanced tab

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

The method the driver uses to encrypt data sent between the driver and the database server. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

Valid Values

0 | 1 | 6

Behavior

If set to 0 (No Encryption), data is not encrypted.

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.

If set to 6 (RequestSSL), the login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established. The SSL protocol used is determined by the setting of the Crypto Protocol Version connection option.

Notes

- This connection option can affect performance.
- Supported by Greenplum 4.2 and higher.

Default

0 (No Encryption)

GUI Tab

Security tab

See also

Performance Considerations on page 379

Extended Column MetaData

Attribute

ExtendedColumnMetaData (ECMD)

Purpose

Determines how the driver returns column metadata when using SQLDescribeCol and SQLColAttribute.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), SQLDescribeCol returns the actual values for Data Type, Column Size, Decimal Digits, and Nullable. SQLColAttribute returns the actual values for:

- SQL_DESC_CATALOG_NAME: *catalog_name*
- SQL_DESC_TABLE_NAME: *table_name*
- SQL_DESC_BASE_COLUMN_NAME: *base_column_name*
- SQL_DESC_LOCAL_TYPE_NAME: *local_type_name*
- SQL_DESC_NULLABLE: *nullable*
- SQL_DESC_AUTO_UNIQUE_VALUE: *auto_unique_value*

If set to 0 (Disabled), SQLDescribeCol returns the Data Type, Column Size, and Decimal Digits for the column. The value SQL_NULLABLE_UNKNOWN is returned for Nullable. SQLColAttribute returns the following attribute values:

- SQL_DESC_CATALOG_NAME: empty string
- SQL_DESC_TABLE_NAME: empty string
- SQL_DESC_BASE_COLUMN_NAME: empty string
- SQL_DESC_LOCAL_TYPE_NAME: empty string
- SQL_DESC_NULLABLE: SQL_NULLABLE_UNKNOWN
- SQL_DESC_AUTO_UNIQUE_VALUE: SQL_FALSE

Default

0 (Disabled)

GUI Tab

Advanced tab

Failover Granularity

Attribute

FailoverGranularity (FG)

Purpose

Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection.

This option applies only when Failover Mode is set to 1 (Extended Connection) or 2 (Select).

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

0 | 1 | 2 | 3

Behavior

If set to 0 (Non-Atomic), the driver continues with the failover process and posts any errors on the statement on which they occur.

If set to 1 (Atomic) the driver fails the entire failover process if an error is generated as the result of anything other than executing and repositioning a Select statement. If an error is generated as a result of repositioning a result set to the last row position, the driver continues with the failover process, but generates a warning that the Select statement must be reissued.

If set to 2 (Atomic Including Repositioning), the driver fails the entire failover process if any error is generated as the result of restoring the state of the connection or the state of work in progress.

If set to 3 (Disable Integrity Check), the driver does not verify that the rows that were restored during the failover process match the original rows. This value applies only when Failover Mode is set to 2 (Select).

Default

0 (Non-Atomic)

GUI Tab

Failover tab

Failover Mode

Attribute

FailoverMode (FM)

Purpose

Specifies the type of failover method the driver uses.

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

0 | 1 | 2

Behavior

If set to 0 (Connection), the driver provides failover protection for new connections only.

If set to 1 (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress.

If set to 2 (Select), the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed.

Default

0 (Connection)

GUI Tab

Failover tab

Failover Preconnect

Attribute

FailoverPreconnect (FP)

Purpose

Specifies whether the driver tries to connect to the primary and an alternate server at the same time.

This attribute applies only when Failover Mode is set to 1 (Extended Connection) or 2 (Select) and at least one alternate server is specified.

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

Valid Values

0 | 1

Behavior

If set to 0 (Disabled), the driver tries to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection. This value provides the best performance, but your application typically experiences a short wait while the failover connection is attempted.

If set to 1 (Enabled), the driver tries to connect to the primary and an alternate server at the same time. This can be useful if your application is time-sensitive and cannot absorb the wait for the failover connection to succeed.

Default

0 (Disabled)

GUI Tab

Failover tab

Fetch RefCursors

Attribute

FetchRefCursors (FRC)

Purpose

Determines whether the driver returns refcursors from stored procedures as results sets.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver returns refcursors from stored procedures as result sets. The driver fetches all the data from the refcursor and then closes the refcursor. If a stored procedure returns multiple refcursors, the driver generates multiple result sets, one for each refcursor returned.

If set to 0 (Disabled), the driver returns the cursor name for refcursors. The application must fetch the actual data from the refcursor using the cursor name and must close the cursor before additional processing can be done on the statement. The application must close the cursor regardless of whether it actually fetches data from the cursor.

Default

1 (Enabled)

GUI Tab

Advanced tab

Fetch TSWTZ as Timestamp

Attribute

FetchTSWTZasTimestamp (FTSWTZAT)

Purpose

Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver returns column values with the timestamp with time zone data type as the ODBC type SQL_TYPE_TIMESTAMP. The time zone information in the fetched value is truncated. Use this value if your application needs to process values the same way as TIMESTAMP columns.

If set to 0 (Disabled), the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_VARCHAR. Use this value if your application requires the time zone information in the fetched value.

Default

0 (Disabled)

GUI Tab

Advanced tab

Fetch TWFS as Time

Attribute

FetchTWFSasTime (FTWFSAT)

Purpose

Determines whether the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME or SQL_TYPE_TIMESTAMP.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME. The fractional seconds portion of the value is truncated.

If set to 0 (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP. The fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp.

Notes

- When returning time with fractional seconds data as SQL_TYPE_TIMESTAMP, the Year, Month and Day parts of the timestamp must be set to zero.

Default

0 (Disabled)

GUI Tab

Advanced tab

GSS Client Library

Attribute

GSSClient (GSSC)

Purpose

The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

The driver uses the path defined by the PATH environment variable for loading the specified client library.

Valid Values

`native` | *`client_library`*

where:

`client_library`

is a GSS client library installed on the client.

Behavior

If set to *`client_library`*, the driver uses the specified GSS client library.

Note: For MIT Kerberos distributions, you must provide a full path to the MIT Library. For example, the 64-bit version for Windows would use the following value: `C:\Program Files\MIT\Kerberos\bin\gssapi64.dll`.

If set to `native`, the driver uses the GSS client for Windows Kerberos. All other users must provide the full path to the library name.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the GSS Client Library option. The GSS Client Library option provides a method for you to specify a library file used to communicate with the Key Distribution Center (KDC). However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.

Default

`native`

GUI Tab

Security tab

Host Name

Attribute

HostName (HOST)

Purpose

The name or the IP address of the server to which you want to connect.

Valid Values

`server_name` | *`IP_address`*

where:

`server_name`

is the name of the server to which you want to connect.

`IP_address`

is the IP address of the server to which you want to connect.

The IP address must be in IPv4 format.

Example

If your network supports named servers, you can specify a server name such as `MainServer`. Or, you can specify an IP address such as `199.226.224.34`..

Default

None

GUI Tab

General tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when SSL encryption is enabled (Encryption Method=1 or 6) and validation is enabled (Validate Server Certificate=1). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values

`host_name | #SERVERNAME#`

where:

`host_name`

is the host name specified in the certificate. Consult your SSL administrator for the correct value.

Behavior

If *`host_name`* is specified, the driver compares the specified host name to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name with the Common Name (CN) part of the certificate. If the values do not match, the connection fails and the driver throws an exception.

If `#SERVERNAME#` is specified, the driver compares the server name that is specified in the connection URL or data source of the connection to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name to the CN part of the certificate's Subject name. If the values do not match, the connection fails and the driver throws an exception. If multiple CN parts are present, the driver validates the host name against each CN part. If any one validation succeeds, a connection is established.

Default

None

Notes

- Supported by Greenplum 4.2 and higher.

GUI Tab

Security tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled or if your database character set is not Unicode. The value you specify must match the database character encoding and the system locale.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values

IANA_code_page

where:

IANA_code_page

is one of the valid values listed in "IANAAppCodePage values" in the *Progress DataDirect for ODBC Drivers Reference*. The value must match the database character encoding and the system locale.

Notes

Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

Advanced tab

Initialization String

Attribute

InitializationString (IS)

Purpose

A SQL command that is issued immediately after connecting to the database to manage session settings.

Valid Values

SQL_command

where:

SQL_command

is a valid SQL command that is supported by the database.

Example

To set the date format on every connection, specify:

```
Set DateStyle='ISO, MDY'
```

Notes

- If the statement fails to execute, the connection fails and the driver reports the error returned from the server.

Default

None

GUI Tab

Advanced tab

Keyset Cursor Options

Attribute

KeysetCursorOptions (KCO)

Purpose

Determines which columns are used to comprise the keyset that the driver uses to create the initial keyset on which cursor operations are based. Greenplum does not offer a true row identifier column; the driver instead uses two hidden system columns provided by the Greenplum database, ctid and gp_segment_id. Because the database might reassign these IDs following a Vacuum operation, the driver can be configured to also include other columns to help ensure that data integrity is maintained.

Valid Values

0 | 1

Behavior

If set to 1 - RowID and Searchable Columns (Enabled), the driver uses a combination of every non-LOB column in the Select list and the ctid and gp_segment_id hidden columns to build the keyset. By adding other Select list fields to the keyset, the driver is able to indicate the row cannot be found if the IDs change following a Vacuum operation.

If set to 0 - RowID Columns (Disabled), the driver uses the ctid and gp_segment_id hidden system columns.

Notes

- This option has no effect unless the EnableKeysetCursors (EKC) connection option is enabled.

Default

0

GUI Tab

Advanced tab

Key Password

Attribute

KeyPassword (KP)

Purpose

Specifies the password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1 or 6) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values

key_password

where:

key_password

is the password of a key in the keystore.

Default

None

Notes

- Supported by Greenplum 4.2 and higher.

GUI Tab

Security tab

Keystore

Attribute

Keystore (KS)

Purpose

The name of the directory containing the keystore file to be used when SSL is enabled (Encryption Method=1 or 6) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request. If you do not specify a directory, the current directory is used.

Valid Values

keystore_directory

where:

`keystore_directory`

is the location of the keystore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Keystore option. The Keystore option provides a method for you to specify a keystore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The keystore and truststore files can be the same file.
- Supported by Greenplum 4.2 and higher.

Default

None

GUI Tab

Security tab

Keystore Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when SSL is enabled (Encryption Method=1 or 6) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

Valid Values

keystore_password

where:

`keystore_password`

is the password of the keystore file.

Notes

- The keystore and truststore files may be the same file; therefore, they may have the same password.
- Supported by Greenplum 4.2 and higher.

Default

None

GUI Tab

Security tab

Load Balance Timeout

Attribute

LoadBalanceTimeout (LBT)

Purpose

Specifies the number of seconds to keep inactive connections open in a connection pool. An inactive connection is a database session that is not associated with an ODBC connection handle, that is, a connection in the pool that is not in use by an application.

Valid Values

0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to 0, inactive connections are kept open.

If set to *x*, inactive connections are closed after the specified number of seconds passes.

Notes

- The Min Pool Size option may cause some connections to ignore this value.

Default

0

GUI Tab

Pooling tab

See also

Performance Considerations on page 379

Load Balancing

Attribute

LoadBalancing (LB)

Purpose

Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order.

If set to 0 (Disabled), the driver does not use client load balancing and connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified).

Notes

- This option has no effect unless alternate servers are defined for the Alternate Servers connection option.

Default

0 (Disabled)

GUI Tab

Failover tab

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

The number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error. To override the value that is set by this connection option for an individual connection, set a different value in the SQL_ATTR_LOGIN_TIMEOUT connection attribute using the SQLSetConnectAttr() function.

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that represents a number of seconds.

Behavior

If set to -1, the connection request does not time out. The driver silently ignores the SQL_ATTR_LOGIN_TIMEOUT attribute.

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute.

If set to x, the connection request times out after the specified number of seconds unless the application overrides this setting with the SQL_ATTR_LOGIN_TIMEOUT attribute.

Default

15

GUI Tab

Advanced tab

Max Char Size

Attribute

MaxCharSize (MCS)

Purpose

Specifies the maximum size of columns of type SQL_CHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from 1 to 10485760

When not specified, the actual size of the columns from the database is persisted to the application.

If you specify a value that is not in the specified range, the driver uses the maximum value of the SQL_CHAR data type.

Default

None. The actual size of the columns from the database is persisted to the application.

GUI Tab

Advanced tab

Max Long Varchar Size

Attribute

MaxLongVarcharSize (MLVS)

Purpose

Specifies the maximum size of columns of type SQL_LONGVARCHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from 1 to x

where:

x

is maximum size of the SQL_LONGVARCHAR data type.

Default

None. The actual size of the columns from the database is persisted to the application.

GUI Tab

Advanced tab

Max Pool Size

Attribute

MaxPoolSize (MXPS)

Purpose

The maximum number of connections allowed within a single connection pool. When the maximum number of connections is reached, no additional connections can be created in the connection pool.

Valid Values

An integer from 1 to 65535

For example, if set to 20, the maximum number of connections allowed in the pool is 20.

Notes

- This connection option can affect performance.

Default

100

GUI Tab

Pooling tab

See also

Performance Considerations on page 379

Min Pool Size

Attribute

MinPoolSize (MNPS)

Purpose

The minimum number of connections that are opened and placed in a connection pool, in addition to the active connection, when the pool is created. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

Valid Values

0 | *x*

Behavior

If set to 0, no connections are opened in addition to the current existing connection.

If set to *x*, the start-up number of connections in the pool is 5 in addition to the current existing connection.

Notes

- This connection option can affect performance.

Example

If set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

Default

0

GUI Tab

Pooling tab

See also

Performance Considerations on page 379

Max Varchar Size

Attribute

MaxVarcharSize (MVS)

Purpose

Specifies the maximum size of columns of type SQL_VARCHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from 1 to *x*

where:

x

is maximum size of the SQL_VARCHAR data type.

Default

None. The actual size of the columns from the database is persisted to the application.

GUI Tab

Advanced tab

OpenSSLConfigFile

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

Specifies the absolute path to the configuration file required to load the FIPS provider when the driver is configured to use OpenSSL with FIPS provider for TLS/SSL encryption (`EnableFIPS=1`).

Valid Values

`fips_config_file`

where:

`fips_config_file`

is the absolute path to the configuration file. For example: `/opt/Progress/DataDirect/ODBC/lib/openssl.cnf`.

Notes

- The OpenSSLConfigFile option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *`install_dir`*`\drivers\openssl.cnf` (Windows)
- *`install_dir`*`/lib/openssl.cnf` (UNIX/Linux)

OpenSSLProviderPath

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

Specifies the path to the directory that contains the provider library when TLS/SSL encryption is enabled.

Valid Values

`provider_path`

where:

`provider_path`

is the path to the directory that contains the provider library.

Notes

- The OpenSSLProviderPath option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *install_dir*\drivers (Windows)
- *install_dir*/lib (UNIX/Linux)

Password

Attribute

Password (PWD)

Purpose

The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

The port number of the server listener.

Valid Values

port_name

where:

port_name

is the port number of the server listener. Check with your database administrator for the correct number.

Default

5432

GUI Tab

General tab

Query Timeout

Attribute

QueryTimeout (QT)

Purpose

The number of seconds for the default query timeout for all statements that are created by a connection. To override the value set by this connection option for an individual statement, set a different value in the SQL_ATTR_QUERY_TIMEOUT statement attribute on the SQLSetStmtAttr() function.

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that specifies a number of seconds.

Behavior

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute.

If set to *x*, all queries time out after the specified number of seconds unless the application overrides this value by setting the SQL_ATTR_QUERY_TIMEOUT attribute.

Default

0

GUI Tab

Advanced tab

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another.

An error message or warning can occur if an ODBC call causes a conversion error, or if an error occurs during code page conversions to and from the database or to and from the application. The error or warning generated is `Code page conversion error encountered`. In the case of parameter data conversion errors, the driver adds the following sentence: `Error in parameter` *x*, where *x* is the parameter number. The standard rules for returning specific row and column errors for bulk operations apply.

Valid Values

0 | 1 | 2

Behavior

If set to 0 (Ignore Errors), the driver substitutes 0x1A for each character that cannot be converted and does not return a warning or error.

If set to 1 (Return Error), the driver returns an error instead of substituting 0x1A for unconverted characters.

If set to 2 (Return Warning), the driver substitutes 0x1A for each character that cannot be converted and returns a warning.

Default

0 (Ignore Errors)

GUI Tab

Advanced tab

Service Principal Name

Attribute

ServicePrincipalName (SPN)

Purpose

The service principal name to be used by driver for Kerberos authentication.

Valid Values

`servicePrincipalName`

where:

`servicePrincipalName`

is a valid service principal name.

If unspecified, the value of the Network Address option is used as the service principal name.

Notes

- If Authentication Method is set to 0, the value of the Service Principal Name option is ignored.

Default

None

GUI Tab

Security tab

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

The absolute path for the OpenSSL library file containing the TLS/SSL library to be used by the data source or connection when TLS/SSL is enabled. The TLS/SSL library contains the implementations of TLS/SSL protocols the driver uses for data encryption.

This option allows you to designate a different TLS/SSL library if you encounter issues with the default version or want to use a library that you provide. Common issues that require designating a different library include security vulnerabilities with specific libraries or compatibility issues with your server or application.

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

is the name of the OpenSSL library file containing the TLS/SSL Library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll
```

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the SSLLibName option. The SSLLibName option provides a method for you to specify an OpenSSL library file used for SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.
- This option must be configured if you are using OpenSSL version 3.0.

Default

No default value

GUI Tab

The value for this option is specified as an option-value pair in the Extended Options field on the Advanced tab. For example:

```
SSLLibName=C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll;
```

See Advanced tab for details.

See also

CryptoLibName on page 346

TCP Keep Alive

Attribute

KeepAlive (KA)

Purpose

Specifies whether the driver enables TCPKeepAlive. TCPKeepAlive maintains idle TCP connections by periodically passing packets between the client and server. If either the client or server does not respond to a packet, the connection is considered inactive and is terminated. In addition, TCPKeepAlive prevents valid idle connections from being disconnected by firewalls and proxies by maintaining network activity.

Valid Values

0 | 1

Behavior

If set to 0 (Disabled), the driver does not enable TCPKeepAlive.

If set to 1 (Enabled), the driver enables TCPKeepAlive.

Default

0 (Disabled)

GUI Tab

Advanced tab

Transaction Error Behavior

Attribute

TransactionErrorBehavior (TEB)

Purpose

Determines how the driver handles errors that occur within a transaction. When an error occurs in a transaction, the Greenplum server does not allow any operations on the connection except for rolling back the transaction.

Valid Values

0 | 1

Behavior

If set to 0 (None), the driver does not roll back the transaction when an error occurs. The application must handle the error and roll back the transaction. Any operation on the statement other than a rollback results in an error.

If set to 1 (Rollback Transaction), the driver rolls back the transaction when an error occurs. In addition to the original error message, the driver posts an error message indicating that the transaction has been rolled back.

Default

1 (Rollback Transaction)

GUI Tab

Advanced tab

Truststore

Attribute

Truststore (TS)

Purpose

The directory that contains the truststore file and the truststore file name to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication. If you do not specify a directory, the current directory is used.

Valid Values

*truststore_directory**filename*

where:

truststore_directory

is the directory where the truststore file is located

filename

is the file name of the truststore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Truststore option. The Truststore option provides a method for you to specify a truststore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The truststore and keystore files may be the same file.

- Supported by Greenplum 4.2 and higher.

Default

None

GUI Tab

Security tab

Truststore Password

Attribute

TruststorePassword (TSP)

Purpose

Specifies the password that is used to access the truststore file when SSL is enabled (Encryption Method=1 or 6) and server authentication is used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

Valid Values

truststore_password

where:

truststore_password

is a valid password for the truststore file.

Notes

- The truststore and keystore files may be the same file; therefore, they may have the same password.
- Supported by Greenplum 4.2 and higher.

Default

None

GUI Tab

Security tab

User Name

Attribute

LogonID (UID)

Purpose

The default user ID that is used to connect to your database. Your ODBC application may override this value or you may override it in the logon dialog box or connection string.

Valid Values

userid

where:

userid

is a valid user ID with permissions to access the database.

Default

None

GUI Tab

Advanced tab

Security tab

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (Encryption Method=1). When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate returned from the server even if the issuer is not a trusted CA is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

If set to 0 (Disabled), the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options.

Notes

- Truststore information is specified using the TrustStore and TrustStorePassword options.
- Supported by Greenplum 4.2 and higher.

Default

1 (Enabled)

GUI Tab

Security tab

Unbounded Numeric Precision

Attribute

UnboundedNumericPrecision (UNP)

Purpose

Specifies the precision for unbounded NUMERIC columns when they are described within the column, parameter, result set, or table metadata. Executing aggregation operations (for example, sum or avg) on bounded NUMERIC columns often results in the server returning the aggregate column as an unbounded NUMERIC column. When this occurs, this option defines the precision for the aggregate column.

Valid Values

A positive integer from 1 to 1000

Default

1000

GUI Tab

Advanced tab

Unbounded Numeric Scale

Attribute

UnboundedNumericScale (UNS)

Purpose

Specifies the scale for unbounded NUMERIC columns when they are described within the column, parameter, result set, or table metadata. Executing aggregation operations (for example, sum or avg) on bounded NUMERIC columns often results in the server returning the aggregate column as an unbounded NUMERIC column. When this occurs, this option defines the scale for the aggregate column.

Valid Values

A positive integer from 1 to 998

Notes

- The driver returns the scale specified in this option for the affected columns regardless of the number of decimal digits in a value. If a value contains fewer digits to the right of the decimal than the specified scale, the remaining digits are automatically returned as padded 0s. For example, if your scale is set to 6 and your value is 22.22, the value returned is 22.220000.

Default

6

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.

Important: When Kerberos is enabled, if the database user name differs from the domain user name, you are required to pass the database user name via the User Name (LogonID) option in the datasource or connection string.

Valid Values

userid

where:

userid

is a valid user ID with permissions to access the database.

Default

None

GUI Tab

Security tab

XML Describe Type

Attribute

XMLDescribeType (XDT)

Purpose

The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See Using the XML Data Type on page 381 for further information about the XML data type.

Valid Values

-4 | -10

Behavior

If set to -4 (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to -10 (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

-10

GUI Tab

Advanced tab

Performance Considerations

The following connection options can enhance driver performance.

Application Using Threads (ApplicationUsingThreads): The driver coordinates concurrent database operations (operations from different threads) by acquiring locks. Although locking prevents errors in the driver, it also decreases performance. If your application does not make ODBC calls from different threads, the driver has no reason to coordinate operations. In this case, the ApplicationUsingThreads attribute should be disabled (set to 0).

Note: If you are using a multi-threaded application, you must enable the Application Using Threads option.

Batch Mechanism (BatchMechanism): Setting BatchMechanism to 2 (MultiRowInsert) or 3 (Copy) provides significant performance gains over 1 (SingleRowInsert) when executing batch inserts:

- When `BatchMechanism=2`, the driver executes a single insert statement for all the rows contained in a parameter array. If the size of the insert statement exceeds the available buffer memory of the server, the driver executes multiple statements.
- When `BatchMechanism=3`, the driver uses the Greenplum COPY command to insert rows into the target table.

Connection Pooling (Pooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout (LoadBalanceTimeout)**: You can define how long to keep connections in the pool. The time that a connection was last used is compared to the current time and, if the timespan exceeds the value of the Load Balance Timeout option, the connection is destroyed. The Min Pool Size option can cause some connections to ignore this value.
- **Connection Reset (ConnectionReset)**: Resetting a re-used connection to the initial configuration settings impacts performance negatively because the connection must issue additional commands to the server.
- **Max Pool Size (MaxPoolSize)**: Setting the maximum number of connections that the pool can contain too low might cause delays while waiting for a connection to become available. Setting the number too high wastes resources.
- **Min Pool Size (MinPoolSize)**: A connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size, if one has been specified. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

Failover Mode (FailoverMode): Although high availability that replays queries after a failure provides increased levels of protection, it can adversely affect performance because of increased overhead.

Data Types

The following table shows how the Greenplum data types are mapped to the standard ODBC data types.

Table 42: Greenplum Data Types

Greenplum	ODBC
Bigint	SQL_BIGINT
Bigserial	SQL_BIGINT
Bit[12]	SQL_BIT
Bit varying	SQL_VARBINARY
Boolean	SQL_BIT
Bytea	SQL_VARBINARY
Character	SQL_CHAR
Character varying	SQL_VARCHAR
Citext[13,14]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Double Precision	SQL_DOUBLE
Float	SQL_REAL
Integer	SQL_INTEGER
Money	SQL_DOUBLE
Name	SQL_VARCHAR
Numeric[15]	SQL_NUMERIC
Real	SQL_REAL
Serial	SQL_INTEGER
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR

[12] Bit maps to SQL_BIT when the length for the bit is 1. If the length is greater than 1, the driver maps the column to SQL_BINARY.

[13] The Citext data type behaves the same as the Text data type, except that it is case-insensitive. The select operations performed on Citext columns return case-insensitive results.

[14] Supported for Greenplum versions 5.3 and higher.

[15] Numeric maps to SQL_NUMERIC if the precision of the Numeric is less than or equal to 38. If the precision is greater than 38, the driver maps the column to SQL_VARCHAR.

Greenplum	ODBC
Time[16]	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Timestamp with timezone [17]	SQL_VARCHAR
Tinyint	SQL_SMALLINT
Wchar	SQL_CHAR
Wvarchar	SQL_VARCHAR

See Retrieving Data Type Information on page 47 for more information about data types.

Using the XML Data Type

By default, Greenplum returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the attribute XMLDescribeType (XDT) to SQL_LONGVARBINARY (-10) and bind the data as SQL_C_BINARY.

Unicode Support

The Greenplum Wire Protocol driver automatically determines whether the Greenplum database is a Unicode database.

Advanced Features

The driver supports the following advanced features:

- Failover
- Connection pooling
- Security

Failover

The driver supports failover and its related connection options. Failover connection options are located on the Failover tab of the driver Setup dialog box. See Using Failover on page 55 for a general description of failover and its implementation.

[16] Time mapping changes based on the setting of the Fetch TWFS as Time option.
[17] Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.

Connection Pooling

The driver supports connection pooling and its related connection options. Connection pooling connection options are located on the Pooling tab of the driver Setup dialog box. See Using DataDirect Connection Pooling on page 73 for a general description of connection pooling and its implementation.

Security

The driver supports authentication and encryption. Security connection options are located on the Security tab of the driver Setup dialog box. See Using Security on page 63 for a general description of security and its implementation.

User-defined Functions' Results

Greenplum provides functionality to create user-defined functions. Greenplum does not define a call mechanism for invoking a user-defined function. User-defined functions must be invoked via a SQL statement.

For example, a function defined as:

```
CREATE table foo (intcol int, varcharcol varchar(123))
CREATE or REPLACE FUNCTION insertFoo
(IN idVal int, IN nameVal varchar) RETURNS void
  AS $$
    insert into foo values ($1, $2);
  $$
  LANGUAGE SQL;
```

must be invoked natively as:

```
SELECT * FROM insertFoo(100, 'Mark')
```

even though the function does not return a value or results. The Select SQL statement returns a result set that has one column named insertFoo and no row data.

The Greenplum Wire Protocol driver supports invoking user-defined functions using the ODBC call Escape. The previously described function can be invoked using:

```
{call insertFoo(100, 'Mark')}
```

Greenplum functions return data from functions as a result set. If multiple output parameters are specified, the values for the output parameters are returned as columns in the result set. For example, the function defined as:

```
CREATE or REPLACE FUNCTION addValues(in v1 int, in v2 int)
  RETURNS int
  AS $$
    SELECT $1 + $2;
  $$
  LANGUAGE SQL;
```

returns a result set with a single column of type SQL_INTEGER, whereas the function defined as:

```
CREATE or REPLACE FUNCTION selectFooRow2
  (IN idVal int, OUT id int, OUT name varchar)
  AS $$
    select intcol, varcharcol from foo where intcol = $1;
  $$
  LANGUAGE SQL
```

returns a result set that contains two columns, a SQL_INTEGER id column and a SQL_VARCHAR name column.

In addition, when calling Greenplum functions that contain output parameters, the native syntax requires that the output parameter values be omitted from the function call. This, in addition to output parameter values being returned as a result set, makes the Greenplum behavior of calling functions different from most other databases.

The Greenplum Wire Protocol driver provides a mechanism that makes the invoking of functions more consistent with how other databases behave. In particular, the Greenplum Wire Protocol driver allows parameter markers for output parameters to be specified in the function argument list when the Escape call is used. The driver allows buffers to be bound to these output parameters. When the function is executed, the output parameters are removed from the argument list sent to the server. The driver extracts the output parameter values from the result set returned by the server and updates the bound output parameter buffers with those values. For example, the function `selectFooRow2` described previously can be invoked as:

```
sql = L"{call selectFooRow2(?, ?, ?)}";
retVal = SQLPrepare(hPrepStmt, sql, SQL_NTS);
retVal = SQLBindParameter(
  hPrepStmt, 1, SQL_PARAM_INPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf, 0, &idInd);
retVal = SQLBindParameter(
  hPrepStmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf2, 4, &idInd2);
retVal = SQLBindParameter(
  hPrepStmt, 3, SQL_PARAM_INPUT, SQL_C_WCHAR,
  SQL_VARCHAR, 30, 0, &nameBuf, 123, &nameInd);
retVal = SQLExecute(hPrepStmt);
```

The values of the id and name output parameters are returned in the `idBuf2` and `nameBuf` buffers.

If output parameters are bound to a function call, the driver returns the output parameters in the bound buffers. An error is returned if the number of output parameters bound when the function is executed is less than the number of output parameters defined in the function. If no output parameters are bound to a function call, the driver returns the output parameters as a result set.

Greenplum can also return results from a function as a refcursor. There can be, at most, one refcursor per result; however, a function can return multiple results where each result is a refcursor. A connection option defines how the driver handles refcursors. See Fetch RefCursors on page 354 for details about this option.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See Persisting a Result Set as an XML Data File on page 48 for details about implementation.

Isolation and Lock Levels Supported

Greenplum supports isolation level 0 (read uncommitted), level 1 (read committed), 2 (Repeatable read), and level 3 (serializable). Greenplum supports record-level locking.

Refer to "Locking and isolation levels" in the *Progress DataDirect for ODBC Drivers Reference* for details.

SQL Support

The driver supports the core SQL grammar.

ODBC Conformance Level

The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions.

In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLTablePrivileges

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported API functions.

Number of Connections and Statements Supported

The Greenplum Wire Protocol driver supports multiple connections and multiple statements per connection.

Using Arrays of Parameters

Greenplum supports returning a set of output parameters or return values, but no ODBC standard method exists for returning arrays of output parameters or return values. If the call Escape is used to invoke a function that returns a set of output parameters and buffers are bound for those output parameters, the Greenplum Wire Protocol driver places the first set of output parameters in the bound buffers. If no output parameters are bound for functions that return a set of results or output parameters, the driver returns a result set with a row for each set of output parameters.

The Impala Wire Protocol Driver

The DataDirect Connect XE *for* ODBC and DataDirect Connect64 XE *for* ODBC for Impala™ Wire Protocol driver support Cloudera Impala database servers.

The Impala server is compatible with data stored in a variety of file formats. In addition, Impala can work with data stored in other systems through the use of storage handlers. The file formats and storage handlers in use should work seamlessly with the Impala driver. The Impala Wire Protocol driver is formally certified with the most prevalent file formats and storage handlers:

- Certified File Formats:
  - Parquet
  - TextFile
- Certified Storage Handlers:
  - HBase

For the latest support information, visit the DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

The driver is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect driver for the file name of the driver.

Driver Requirements

The driver has no client requirements.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 390 and Connection Option Descriptions on page 392 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

UNIX® On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration on page 27 for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Connection Option Descriptions on page 392 lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI (Impala)

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. UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a data source for Impala:

1. Start the ODBC Administrator:
   - On Windows, start the ODBC Administrator by selecting its icon from the Datadirect Connect 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.

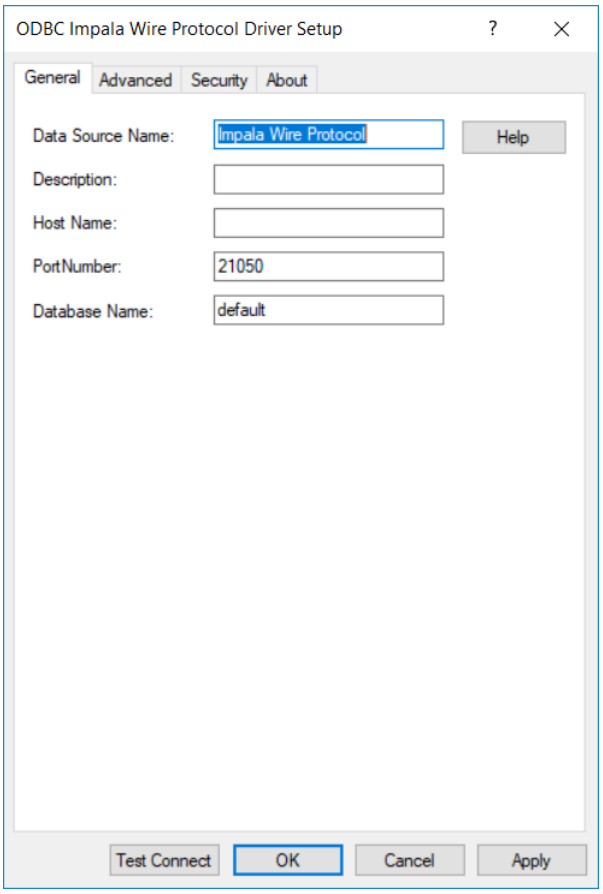
     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** 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.

The General tab of the Setup dialog box appears by default.

Figure 35: General tab



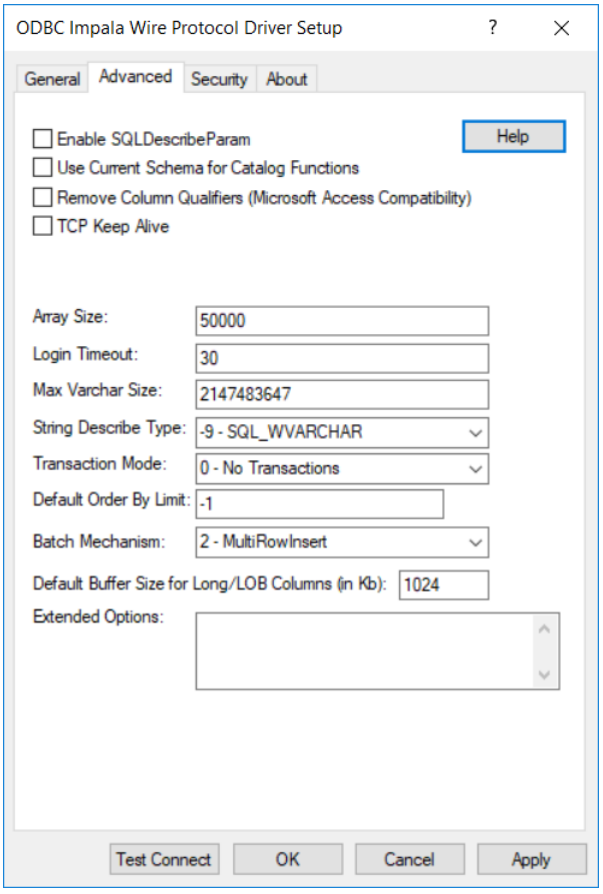
Note: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3. On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name on page 398	None
Description on page 400	None
Host Name on page 403	None
Port Number on page 410	None
Database on page 398	default

4. Optionally, click the **Advanced** tab to specify additional data source settings.

Figure 36: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Enable SQLDescribeParam on page 401	Disabled
Use Current Schema for Catalog Functions on page 417	Disabled
Remove Column Qualifiers on page 411	Disabled
TCP Keep Alive on page 414	Disabled
Array Size on page 394	50000
Login Timeout on page 407	30
Max Varchar Size on page 407	2147483647
String Describe Type on page 414	-9 - SQL_WVARCHAR
Transaction Mode on page 415	0 - No Transactions
Default Order By Limit on page 399	-1 (Disabled)
Batch Mechanism on page 395	2 - MultiRowInsert
Default Buffer Size for Long/LOB Columns (in Kb) on page 399	1024

Extended Options: Type a semi-colon separated list of connection options and their values. Use this configuration option to set the value of undocumented connection options that are provided by Progress DataDirect customer support. You can include any valid connection option in the Extended Options string, for example:

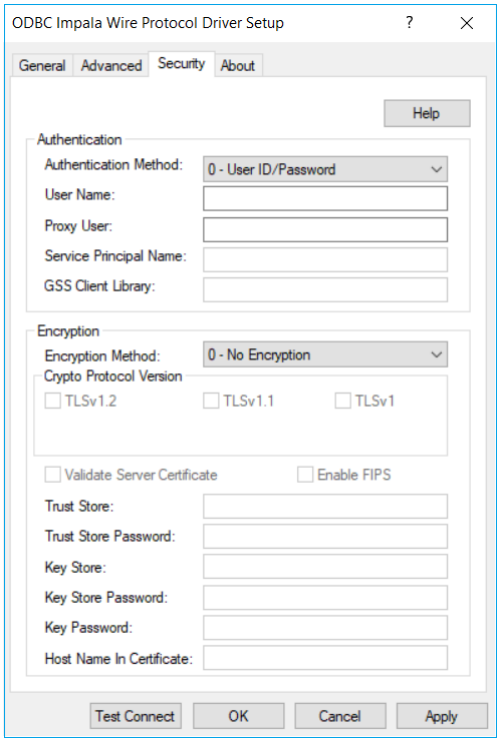
`Database=Server1; UndocumentedOption1=value[;UndocumentedOption2=value;]`

If the Extended Options string contains option values that are also set in the setup dialog or data source, the values of the options specified in the Extended Options string take precedence. However, connection options that are specified on a connection string override any option value specified in the Extended Options string.

Note: Do not specify the Extended Options configuration option in a connection string, or the driver will return an error. Instead, applications should specify the individual undocumented connection options in the connection string.

5. Optionally, click the **Security** tab to specify security settings.

Figure 37: Security tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options:Security	Default
Authentication Method on page 394	0 - User ID/Password
User Name on page 417	None
Proxy User on page 410	None
Service Principal Name on page 411	None
GSS Client Library on page 402	native
Encryption Method on page 402	0 - No Encryption
Crypto Protocol Version on page 396	TLSv1.2,TLSv1.1,TLSv1,SSLv3
Validate Server Certificate on page 418	1 - Enabled
Enable FIPS on page 119	Disabled
Truststore on page 416	None
Trust Store Password on page 415	None
Key Store on page 405	None

Connection Options:Security	Default
Keystore Password on page 406	None
Key Password on page 405	None
Host Name In Certificate on page 404	None

6. 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 (Impala) on page 391) for details). Note that the information you enter in the logon dialog box during a test connect is not saved.
   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

   Note: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

7. Click **OK** or **Cancel**. If 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.

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 on page 392 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 Impala is:

```
DSN=Accounting;UID=JOHN;PWD=XYZZY
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Impala.dsn;UID=JOHN;PWD=XYZZY
```

A DSN-less connection string must provide all necessary connection information:

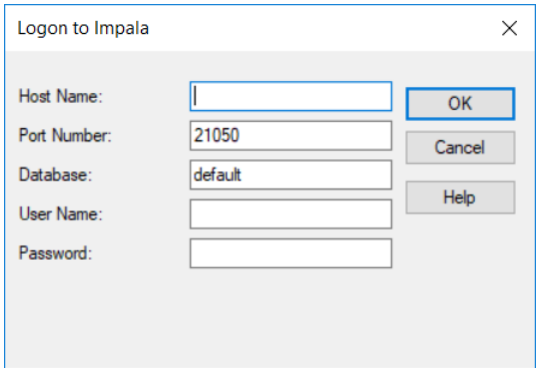
```
DRIVER={DataDirect 7.1 Impala Wire Protocol};HOST=server1;PORT=21050;UID=JOHN;PWD=XYZZY;
```

Using a Logon Dialog Box (Impala)

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.

Note: A user name and password are not required to connect to Impala.

Figure 38: Logon to Impala dialog box



Note: To configure a standard connection, complete the first two fields and skip to Step 4.

In this dialog box, provide the following information:

1. In the Host field, type either the name or the IP address of the server to which you want to connect.
2. In the Port Number field, type the port number that your Impala server is listening on. Check with your Impala administrator for the correct number.
3. In the Database field, type the name of the Impala database. The database must exist, or the connection attempt will fail.
4. If required, type your logon ID In the User Name field.
5. If required, type your password In the Password field.
6. Click **OK** to log on to the Impala server you specified and to update the values in the Registry.

Note: The User Name and Password fields are not used at this time to connect to Impala Server.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Array Size

Attribute

ArraySize (AS)

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.

The following table lists the connection string attributes supported by the driver.

Table 43: Attribute Names for the Driver for Impala

Attribute (Short Name)	Default
ArraySize (AS)	50000
AuthenticationMethod (AM)	0 (User ID/Password)
BatchMechanism (BM)	2 (MultiRowInsert)
CryptoLibName (CLN)	Empty string
CryptoProtocolVersion (CPV)	`TLSv1.2,TLSv1.1,TLSv1,SSLv3`
Database (DB)	default
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
DefaultOrderByLimit (DOBL)	-1 (Disabled)
Description (n/a)	None
EnableDescribeParam (EDP)	0 (Disabled)
EnableFIPS (EF)	Disabled
EncryptionMethod (EM)	0 (No Encryption)
GSSClient (GSSC)	native
HostName (HOST)	None
HostNameInCertificate (HNIC)	None

Attribute (Short Name)	Default
KeepAlive (KA)	0 (Disabled)
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoginTimeout (LT)	30
LogonID (UID)	None
MaxVarcharSize (MVS)	2147483647
OpenSSLConfigFile (OSSLCNF)	*install_dir*\drivers\openssl.cnf (Windows) *install_dir*/lib/openssl.cnf (UNIX/Linux)
OpenSSLProviderPath (OSSLPP)	*install_dir*\drivers (Windows) *install_dir*/lib (UNIX/Linux)
Password (PWD)	None
PortNumber (PORT)	None
ProxyUser (PU)	None
RemoveColumnQualifiers (RCQ)	0 (Disabled)
ServicePrincipalName (SPN)	None
SSLLibName (SLN)	Empty String
StringDescribeType (SDT)	-9 - SQL_WVARCHAR
TransactionMode (TM)	0 (No Transactions)
Truststore (TS)	None
TruststorePassword (TSP)	None
UseCurrentSchema (UCS)	0 (Disabled)
ValidateServerCertificate (VSC)	1 (Enabled)

Array Size

Attribute

ArraySize (AS)

Purpose

The number of cells the driver retrieves from a server for a fetch. When executing a fetch, the driver divides the value specified by the number columns in a particular table to determine the number of rows to retrieve. By determining the fetch size based on the number of cells, the driver can avoid out of memory errors when fetching from tables containing a large number of columns while continuing to provide improved performance when fetching from tables containing a small number of columns.

Valid Values

x

where:

x

is a positive integer specifying the number of cells the driver retrieves for a fetch.

Notes

- You can improve performance by increasing the value specified for this option; however, if the number of cells specified exceeds the available buffer memory for the server, an out of memory error will be returned. If you receive this error, decrease the value specified until fetches are successfully executed.
- This connection option can affect performance.

Default

50000

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 418 for details.

Authentication Method

Attribute

AuthenticationMethod (AM)

Purpose

Specifies the method the driver uses to authenticate the user to the server when a connection is established. If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Valid Values

0 | 4 | -1

Behavior

If set to 0 (User ID/Password), the driver sends the user ID in clear text and an encrypted password to the server for authentication.

If set to 4 (Kerberos Authentication), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

If set to -1 (No Authentication), the driver sends the user ID and password in clear text to the server for authentication.

Default

0 (User ID/Password)

GUI Tab

Security tab

Batch Mechanism

Attribute

BatchMechanism (BM)

Purpose

Determines the mechanism that is used to execute batch operations.

Valid Values

1 | 2

Behavior

If set to 1 (SingleInsert), the driver executes an insert statement for each row contained in a parameter array. Select this setting if you are experiencing out-of-memory errors when performing batch inserts.

If set to 2 (MultiRowInsert), the driver attempts to execute a single insert statement for all the rows contained in a parameter array. If the size of the insert statement exceeds the available buffer memory of the driver, the driver executes multiple statements. Select this setting for substantial performance gains when performing batch inserts.

Default

2 (MultiRowInsert)

Notes

- This connection option can affect performance.

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 418 for details.

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

Specifies a comma-separated list of the cryptographic protocols to use when SSL is enabled using the Encryption Method connection option (EncryptionMethod=1). When multiple protocols are specified, the driver uses the highest version supported by the server. If none of the specified protocols are supported by the database server, behavior is determined by the setting of the EncryptionMethod connection option.

Valid Values

cryptographic_protocol [[, *cryptographic_protocol*]...]

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.2 | TLSv1.3

Example

If your security environment is configured to use TLSv1.2 and TLSv1.3, specify the following values:

```
CryptoProtocolVersion=TLSv1.2, TLSv1.3
```

Notes

- This option is ignored if Encryption Method is set to 0 - No Encryption.
- Consult your database administrator concerning the data encryption settings of your server.

Default

TLSv1.2,TLSv1.1

GUI Tab

Security tab

See also

Encryption Method on page 402

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

The absolute path for the OpenSSL library file containing the cryptographic library to be used by the data source or connection when TLS/SSL is enabled. The cryptograpic library contains the implementations of cryptographic algorithms the driver uses for data encryption.

This option allows you to designate a different cryptographic library if you encounter issues with the default version or want to use a library that you provide. Common issues that require designating a different library include security vulnerabilities with specific libraries or compatibility issues with your server or application.

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

is the name of the OpenSSL library file containing the cryptographic library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll
```

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the CryptoLibName option. The CryptoLibName option provides a method for you to specify a cryptographic library file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.
- This option must be configured if you are using OpenSSL version 3.0.

Default

Empty string

GUI Tab

The value for this option is specified as an option-value pair in the Extended Options field on the Advanced tab. For example:

```
CryptoLibName=C:\Program Files\Progress\DataDirect\ODBC\drivers\ddopenssl30.dll;
```

See Advanced tab for details.

See also

SSLLibName on page 412

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

Database

Attribute

Database (DB)

Purpose

Specifies the name of the Impala database. The database must exist, or the connection attempt will fail.

Valid Values

database_name

where:

database_name

is the name of the Impala database.

Default

default

GUI Tab

General tab

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute

DefaultLongDataBuffLen (DLDBL)

Purpose

The maximum length of data (in KB) the driver can fetch from long columns in a single round trip and the maximum length of data that the driver can send using the SQL_DATA_AT_EXEC parameter.

Valid Values

An integer in multiples of 1024

The value must be in multiples of 1024 (for example, 1024, 2048). You need to increase the default value if the total size of any Long data exceeds 1 MB. This value is multiplied by 1024 to determine the total maximum length of fetched data. For example, if you enter a value of 2048, the maximum length of data would be 1024 x 2048, or 2097152 (2 MB).

Notes

- This connection option can affect performance.

Default

1024

GUI tab

Advanced tab

See Also

See Performance Considerations on page 418 for details.

Default Order By Limit

Attribute

DefaultOrderByLimit (DOBL)

Purpose

Specifies the maximum number of rows returned when a SQL statement containing an ORDER BY clause is executed. Cloudera Impala requires statements containing the ORDER BY clause to limit the number of rows returned. This option allows these statements to return a result set without specifying a limit in the application..

Valid Values

-1 | *x*

where:

x

is a positive integer that represents maximum number of rows returned.

Behavior

If set to -1 (disabled), there is no default limit the number of rows returned by a statement containing an ORDER BY clause. The application must limit the number of rows returned by SQL statements that contain an ORDER BY clause, or Impala will return an error.

If set to *x*, the number of rows returned by a SQL statement contining an ORDER BY clause are limited to the specified number of rows for the session. To override this value, specify a new value in a LIMIT clause in the statement that is being executed.

Default

-1 (Disabled)

GUI Tab

Advanced tab

Description

Attribute

Description (n/a)

Purpose

Specifies an optional long description of a data source. This description is not used as a runtime connection attribute, but does appear in the ODBC.INI section of the Registry and in the odbc.ini file.

Valid Values

`string`

where:

`string`

is a description of a data source.

Default

None

GUI Tab

General tab

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled.

Valid Values

`0` | `1`

Behavior

If set to `0`, the OpenSSL library uses cryptographic algorithms from the default provider.

If set to `1`, the OpenSSL library uses cryptographic algorithms from the FIPS provider.

Notes

- The FIPS provider is supported only on the following platforms: Windows 64-bit, Linux 64-bit, and AIX 64-bit. On the other platforms, the driver uses the default provider of the OpenSSL 3.5 library.
- Do not set the Truststore Password (TruststorePassword) connection option when using the FIPS provider. The truststore password uses the PKCS12KDF algorithm, which is not an approved FIPS algorithm. Hence, it must not be specified when using the FIPS provider.
- For using the FIPS and default providers, the certificates must be encrypted with the OpenSSL 3.5-compliant cryptographic algorithms. See "Generating TLS/SSL certificates with OpenSSL 3.5-compliant algorithms" for more information.

Default

`0`

Enable SQLDescribeParam

Attribute

EnableDescribeParam (EDP)

Purpose

Determines whether the driver uses the SQLDescribeParam function, which describes parameters as a data type of SQL_VARCHAR with a length of 255 for statements.

Valid Values

0 | 1

Behavior

If set to 1 (enabled), the SQLDescribeParam function describes parameters as a data type of SQL_VARCHAR with a length of 255 for statements.

If set to 0 (disabled), the SQLDescribeParam function returns the standard ODBC error IM001.

Default

0 (Disabled)

GUI tab

Advanced tab

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

The method the driver uses to encrypt data sent between the driver and the database server. If the specified encryption method is not supported by the database server, the connection fails and the driver returns an error.

Valid Values

0 | 1

Behavior

If set to 0 (No Encryption), data is not encrypted.

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.

Notes

- This connection option can affect performance.

Default

0 (No Encryption)

GUI Tab

Security tab

See also

Performance Considerations on page 418

GSS Client Library

Attribute

GSSClient (GSSC)

Purpose

The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

The driver uses the path defined by the PATH environment variable for loading the specified client library.

Valid Values

`native` | *`client_library`*

where:

client_library

is a GSS client library installed on the client.

Behavior

If set to *client_library*, the driver uses the specified GSS client library.

Note: For MIT Kerberos distributions, you must provide a full path to the MIT Library. For example, the 64-bit version for Windows would use the following value: `C:\Program Files\MIT\Kerberos\bin\gssapi64.dll`.

If set to `native`, the driver uses the GSS client for Windows Kerberos. All other users must provide the full path to the library name.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the GSS Client Library option. The GSS Client Library option provides a method for you to specify a library file used to communicate with the Key Distribution Center (KDC). However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.

Default

`native`

GUI Tab

Security tab

Host Name

Attribute

HostName (HOST)

Purpose

The name or the IP address of the server to which you want to connect.

Valid Values

host_name | *IP_address*

where:

hostname

is the name of the Impala server to which you want to connect

IP_address

is the IP address of the server to which you want to connect.

Default

None

GUI Tab

General tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when SSL encryption is enabled (Encryption Method=1) and validation is enabled (Validate Server Certificate=1). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values

`host_name | #SERVERNAME#`

where:

`host_name`

is the host name specified in the certificate. Consult your SSL administrator for the correct value.

Behavior

If *host_name* is specified, the driver compares the specified host name to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name with the Common Name (CN) part of the certificate. If the values do not match, the connection fails and the driver throws an exception.

If #SERVERNAME# is specified, the driver compares the server name that is specified in the connection URL or data source of the connection to the DNSName value of the SubjectAlternativeName in the certificate. If the certificate does not have a SubjectAlternativeName, the driver compares the host name to the CN part of the certificate's Subject name. If the values do not match, the connection fails and the driver throws an exception. If multiple CN parts are present, the driver validates the host name against each CN part. If any one validation succeeds, a connection is established.

Default

None

GUI Tab

Security tab

Key Password

Attribute

KeyPassword (KP)

Purpose

Specifies the password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values

key_password

where:

key_password

is the password of a key in the keystore.

Default

None

GUI Tab

Security tab

Key Store

Attribute

Keystore (KS)

Purpose

The name of the directory containing the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request. If you do not specify a directory, the current directory is used.

Valid Values

keystore_directory

where:

keystore_directory

is the location of the keystore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Keystore option. The Keystore option provides a method for you to specify a keystore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The keystore and truststore files can be the same file.

Default

None

GUI Tab

Security tab

Keystore Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

Valid Values

keystore_password

where:

`keystore_password`

is the password of the keystore file.

Notes

- The keystore and truststore files may be the same file; therefore, they may have the same password.

Default

None

GUI Tab

Security tab

Login Timeout

Attribute

LoginTimeout (LT)

Purpose

Specifies the number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error. To override the value that is set by this connection option for an individual connection, set a different value in the SQL_ATTR_LOGIN_TIMEOUT connection attribute using the SQLSetConnectAttr() function.

Valid Values

-1 | 0 | *x*

where:

x

is a positive integer that represents a number of seconds.

Behavior

If set to -1, the connection request does not time out. The driver silently ignores the SQL_ATTR_LOGIN_TIMEOUT attribute.

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute.

If set to *x*, the connection request times out after the specified number of seconds unless the application overrides this setting with the SQL_ATTR_LOGIN_TIMEOUT attribute.

Default

30

GUI Tab

Advanced tab

Max Varchar Size

Attribute

MaxVarcharSize (MVS)

Purpose

Specifies the maximum size of columns of type SQL_VARCHAR that the driver describes through result set descriptions and catalog functions.

Valid Values

A positive integer from 255 to *x*

where:

x

is maximum size of the SQL_VARCHAR data type.

Default

2147483647

GUI Tab

Advanced tab

OpenSSLConfigFile

Attribute

OpenSSLConfigFile (OSSLCNF)

Purpose

Specifies the absolute path to the configuration file required to load the FIPS provider when the driver is configured to use OpenSSL with FIPS provider for TLS/SSL encryption (`EnableFIPS=1`).

Valid Values

`fips_config_file`

where:

`fips_config_file`

is the absolute path to the configuration file. For example: `/opt/Progress/DataDirect/ODBC/lib/openssl.cnf`.

Notes

- The OpenSSLConfigFile option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *`install_dir`*`\drivers\openssl.cnf` (Windows)
- *`install_dir`*`/lib/openssl.cnf` (UNIX/Linux)

OpenSSLProviderPath

Attribute

OpenSSLProviderPath (OSSLPP)

Purpose

Specifies the path to the directory that contains the provider library when TLS/SSL encryption is enabled.

Valid Values

provider_path

where:

provider_path

is the path to the directory that contains the provider library.

Notes

- The OpenSSLProviderPath option is not available on the setup dialog box. To set a value for it, use the Extended Options connection option, which is available on the Advanced tab of the setup dialog box.

Default

- *install_dir*\drivers (Windows)
- *install_dir*/lib (UNIX/Linux)

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.

Note: Not used to log on to Impala at this time.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

Specifies 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

Proxy User

Attribute

ProxyUser (PU)

Purpose

Specifies the UserID used for impersonation. When impersonation is enabled on the server, this value determines your identity and access rights to files when executing queries. If no value is provided for this option or if impersonation is disabled, you will execute queries as the user who initiated the process.

Impersonation provides a method for administrators to control access to data. Administrators set access rights to files by using HDFS and directory permissions on the server.

Valid Values

userid

where:

userid

is a valid user ID with permissions to access the database.

Default

None

Notes

- Supported in Impala 1.2 and higher.

GUI Tab

Security tab

Remove Column Qualifiers

Attribute

RemoveColumnQualifiers (RCQ)

Purpose

Specifies whether the driver removes 3-part column qualifiers and replaces them with alias.column qualifiers. Microsoft Access executes a Select statement using this syntax when an index is specified on a linked table.

Valid Values

0 | 1

Behavior

If set to 1 (enabled) the driver removes 3-part column qualifiers and replaces them with alias.column qualifiers. Column qualifiers are Microsoft Access compatible in this setting.

If set to 0, the driver does not modify the request.

Notes

- When using the driver with Microsoft Access in creating a linked table, it is highly recommended that you do not specify an index. Specifying an index causes Access to execute a Select statement for each row, which results in very slow performance.

Default

0 (Disabled)

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 418 for details.

Service Principal Name

Attribute

ServicePrincipalName (SPN)

Purpose

The service principal name to be used by driver for Kerberos authentication.

Valid Values

ServicePrincipalName

where:

ServicePrincipalName

is the three-part service principal name registered with the key distribution center (KDC).

You must specify the service principal name using the following format:

Service_Name/*Fully_Qualified_Domain_Name*@*REALM.COM*

where:

Service_Name

is the name of the service hosting the instance. For example, `impala`.

The server automatically generates the service name. Refer to Cloudera Impala documentation for additional information.

Fully_Qualified_Domain_Name

is the fully qualified domain name of the host machine. For example, `yourserver.example.com`.

REALM.COM

is the domain name of the host machine. This part of the value must be specified in upper-case characters. For example, `EXAMPLE.COM`.

Example

The following is an example of a valid service principal name:

```
impala/yourserver.example.com@EXAMPLE.COM
```

Notes

- If unspecified, the value of the Network Address option is used as the service principal name.
- If Authentication Method is set to 0 or -1, the value of the Service Principal Name option is ignored.

Default

None

GUI Tab

Security tab

SSLLibName

Attribute

SSLLibName (SLN)

Purpose

The absolute path for the OpenSSL library file containing the TLS/SSL library to be used by the data source or connection when TLS/SSL is enabled. The TLS/SSL library contains the implementations of TLS/SSL protocols the driver uses for data encryption.

This option allows you to designate a different TLS/SSL library if you encounter issues with the default version or want to use a library that you provide. Common issues that require designating a different library include security vulnerabilities with specific libraries or compatibility issues with your server or application.

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

is the name of the OpenSSL library file containing the TLS/SSL Library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll
```

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the SSLLibName option. The SSLLibName option provides a method for you to specify an OpenSSL library file used for SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The OpenSSL library files provided by Progress combine the cryptographic and TLS/SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLLibName options should be the same. For non-Progress library files, the libraries may use separate files, which would require unique values to be specified.
- This option can be used to designate OpenSSL libraries not installed by the product; however, the drivers are only certified against libraries provided by Progress.
- This option must be configured if you are using OpenSSL version 3.0.

Default

No default value

GUI Tab

The value for this option is specified as an option-value pair in the Extended Options field on the Advanced tab. For example:

```
SSLLibName=C:\Program Files\Progress\DataDirect\ODBC\Drivers\ddopenssl30.dll;
```

See Advanced tab for details.

See also

CryptoLibName on page 396

String Describe Type

Attribute

StringDescribeType (SDT)

Purpose

Specifies whether all string columns are described as SQL_VARCHAR. This connection option affects SQL_Columns, SQLDescribeCol, SQLColAttributes, etc. It does not affect SQLGetTypeInfo.

Valid Values

-10 | -9

Behavior

If set to -10 (SQL_WLONGVARCHAR), all string columns are described as SQL_WLONGVARCHAR.

If set to -9 - (SQL_WVARCHAR), all string columns are described as SQL_WVARCHAR.

Default

-9 - SQL_WVARCHAR

GUI Tab

Advanced tab

TCP Keep Alive

Attribute

KeepAlive (KA)

Purpose

Specifies whether the driver enables TCPKeepAlive. TCPKeepAlive maintains idle TCP connections by periodically passing packets between the client and server. If either the client or server does not respond to a packet, the connection is considered inactive and is terminated. In addition, TCPKeepAlive prevents valid idle connections from being disconnected by firewalls and proxies by maintaining network activity.

Valid Values

0 | 1

Behavior

If set to 0 (Disabled), the driver does not enable TCPKeepAlive.

If set to 1 (Enabled), the driver enables TCPKeepAlive.

Default

0 (Disabled)

GUI Tab

Advanced tab

Transaction Mode

Attribute

TransactionMode (TM)

Purpose

Specifies how the driver handles manual transactions.

Valid Values

0 | 1

Behavior

If set to 1 (Ignore), the data source does not support transactions and the driver always operates in auto-commit mode. Calls to set the driver to manual commit mode and to commit transactions are ignored. Calls to rollback a transaction cause the driver to return an error indicating that no transaction is started. Metadata indicates that the driver supports transactions and the ReadUncommitted transaction isolation level.

If set to 0 (No Transactions), the data source and the driver do not support transactions. Metadata indicates that the driver does not support transactions.

Default

`0 (No Transactions)`

GUI Tab

Advanced tab

Trust Store Password

Attribute

TruststorePassword (TSP)

Purpose

Specifies the password that is used to access the truststore file when SSL is enabled (`Encryption Method=1`) and server authentication is used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

Valid Values

`truststore_password`

where:

truststore_password

is a valid password for the truststore file.

Notes

- The truststore and keystore files may be the same file; therefore, they may have the same password.

Default

None

GUI Tab

Security tab

Truststore

Attribute

Truststore (TS)

Purpose

The directory that contains the truststore file and the truststore file name to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication. If you do not specify a directory, the current directory is used.

Valid Values

*truststore_directory**filename*

where:

truststore_directory

is the directory where the truststore file is located

filename

is the file name of the truststore file.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Truststore option. The Truststore option provides a method for you to specify a truststore file used for TLS/SSL encryption. However, if exposed, the option can be used to specify files that execute malicious or undesirable code. Refer to "Security best practices for ODBC applications" in the *Progress DataDirect for ODBC Drivers Reference* for more information.
- The value specified for this option should be an absolute path to a mounted drive.
- The truststore and keystore files may be the same file.

Default

None

GUI Tab

Security tab

Use Current Schema for Catalog Functions

Attribute

UseCurrentSchema (UCS)

Purpose

Specifies whether results are restricted to the tables and views in the current schema if a catalog function call is made without specifying a schema or if the schema is specified as the wildcard character %. Restricting results to the tables and views in the current schema improves performance of catalog calls that do not specify a schema.

Valid Values

`0 | 1`

Behavior

If set to `1 (Enabled)`, results of catalog function calls are restricted to the tables and views in the current schema.

If set to `0 (Disabled)`, results of catalog function calls are not restricted.

Default

`0 (Disabled)`

GUI Tab

Advanced tab

See Also

See Performance Considerations on page 418 for details.

User Name

Attribute

LogonID (UID)

Purpose

The default user ID that is used to connect to your database.

Valid Values

N/A

GUI Tab

Security tab

Validate Server Certificate

Attribute

ValidateServerCertificate (VSC)

Purpose

Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (Encryption Method=1). When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate returned from the server even if the issuer is not a trusted CA is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

If set to 0 (Disabled), the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options.

Notes

- Truststore information is specified using the TrustStore and TrustStorePassword options.

Default

1 (Enabled)

GUI Tab

Security tab

Performance Considerations

The following connection options can enhance driver performance.

Array Size (ArraySize): To improve throughput, consider increasing the value of Array Size. By increasing the value of Array Size, you increase the number of rows the driver will retrieve from the server for a fetch. In turn, increasing the number of rows that the driver can retrieve reduces the number, and expense, of network round trips. For example, if an application attempts to fetch 100,000 rows, it is more efficient for the driver to retrieve 2000 rows over the course of 50 round trips than to retrieve 500 rows over the course of 200 round trips. Note that improved throughput does come at the expense of increased demands on memory and slower response time. Furthermore, if the fetch size exceeds the available buffer memory of the server, an out of memory error is returned when attempting to execute a fetch. If you receive this error, decrease the value specified until fetches are successfully executed.

Batch Mechanism (BatchMechanism): If your application does not require individual update counts for each statement or parameter set in the batch, then BatchMechanism should be set to 2 (MultiRowInsert). Unlike the native batch mechanism, the multi-row insert mechanism only returns the total number of update counts for batch inserts. Therefore, setting BatchMechanism to MultiRowInsert offers substantial performance gains when performing batch inserts.

Default Buffer Size for Long/LOB Columns (DefaultLongDataBuffLen): To improve performance when your application fetches images, pictures, or long text or binary data, a buffer size can be set to accomodate the maximum size of the data. The buffer size should only be large enough to accommodate the maximum amount of data retrieved; otherwise, performance is reduced by transferring large amounts of data into an oversized buffer. If your application retrieves more than 1 MB of data, the buffer size should be increased accordingly.

Encryption Method (**EncryptionMethod**): Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) that is required to encrypt and decrypt data.

Use Current Schema for Catalog Functions (UseCurrentSchema): If your application needs to access database objects owned only by the current user, then performance can be improved. In this case, the Use Current Schema for Catalog Functions option must be enabled. When this option is enabled, the driver returns only database objects owned by the current user when executing catalog functions. Calls to catalog functions are optimized by grouping queries. Enabling this option is equivalent to passing the Logon ID used on the connect as the Schema Name arguement to catalog functions.

Data Types

The following table shows how the Impala data types are mapped to the standard ODBC data types.

Table 44: Impala Data Types

Impala	ODBC
Bigint	SQL_BIGINT
Boolean	SQL_BIT
Char[18]	SQL_CHAR
Decimal[18]	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_REAL
Int	SQL_INTEGER
Smallint	SQL_SMALLINT
String[19]	SQL_WVARCHAR, SQL_WLONGVARCHAR[20]

[18] Supported with Impala 2.0 and higher.

[20] If the StringDescribeType property is set to *wvarchar* (the default), this data type maps to WVARCHAR. If set to *wlongvarchar*, this data type maps to WLONGVARCHAR.

[19] Maximum of 2 GB

Impala	ODBC
Timestamp	SQL_TYPE_TIMESTAMP
Tinyint	SQL_TINYINT
Varchar[18]	SQL_VARCHAR

Advanced Features

The driver supports the following advanced features:

- Security

Security

The driver supports authentication and data encryption. Security connection options are located on the Security tab of the driver Setup dialog box. See Using Security on page 63 for a general description of security and its implementation. The following security-related information is specific to the Impala Wire Protocol driver.

Apache Sentry

Apache Sentry is a modular security system that enables administrators to control access to data and metadata stored on an Apache Hadoop cluster by defining user roles and permissions. The driver works transparently with Sentry and does not require further configuration. To use Sentry, Kerberos authentication must be enabled, and a Kerberos logon must be provided at connection.

Note: When establishing a connection, the driver attempts to set the user's default database to be used for the session. In environments using Sentry, the user must be granted access to this database; otherwise, the connection will fail.

For more information, refer to the Cloudera Impala documentation at http://cloudera.com/content/cloudera/en/documentation.html.

Materialized Views

Impala supports views but purely as logical objects with no associated storage. As such, there is no support for materialized views in Impala; therefore, the driver does not support materialized views.

Stored Procedures

Impala has no concept of stored procedures. Therefore, they are not supported in the driver.

Unicode Support

The driver is fully Unicode enabled. On UNIX and Linux platforms, the driver supports both UTF-8 and UTF-16. On Windows platforms, the Impala driver supports UCS-2/UTF-16 only.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See UTF-16 Applications on UNIX and Linux on page 101 for related details.

Also, refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for a more detailed explanation of Unicode.

Isolation and Lock Levels Supported

Impala supports isolation level 0 (read uncommitted).

Refer to "Locking and isolation levels" in the *Progress DataDirect for ODBC Drivers Reference* for details.

SQL Support

The driver supports the core SQL grammar and the Cloudera Impala query language.

Refer to the Impala SQL Language Reference for information about using Impala SQL.

Also, see SQL Functionality for the Impala Wire Protocol driver.

ODBC Conformance Level

The driver supports ODBC API Conformance Level 1.

Note: SQLCancel and SQLTransact execute successfully but perform no functions.

Note: SQLStatistics always returns an empty result set.

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported API functions.

Using Arrays of Parameters

By default, the driver supports multi-row inserts for parameterized arrays. For a multi-row insert, the driver attempts to execute a single insert for all the rows contained in a parameter array. If the size of the insert statement exceeds the available buffer memory of the driver, the driver executes multiple statements. This behavior provides substantial performance gains for batch inserts.

The driver modifies the native statement to perform a multi-row insert. Therefore, the default multi-row insert behavior may not be desirable in all scenarios. You can disable this behavior by setting the Batch Mechanism connection option to `1` (SingleInsert). When `BatchMechanism=1`, Impala's single row insert is used to execute batch operations, and an insert statement is executed for each row contained in a parameter array.

Refer to "Designing ODBC applications for performance optimization" in the *Progress DataDirect for ODBC Drivers Reference* for details.

Limitations on Cloudera Impala Functionality

The following restrictions are based on using Cloudera Impala version 1.0:

- No support for transactions
- No support for canceling a running query
- No support for row-level inserts, updates, or deletes
- No support for cursors or scrollable cursors
- No support for prepared statements

For a more complete listing of known issues and limitations for your version of Impala, refer to the Cloudera Impala user documentation:

http://www.cloudera.com/content/support/en/documentation.html

---

Note: Note that Cloudera Impala is not designed for OLTP workloads and does not offer real-time queries or row-level updates. Instead, Impala is designed for batch type jobs over large data sets with high latency. This means that queries such as "`SELECT * FROM mytable`" return quickly. However, other SELECT statements are much slower.

The Driver for the Teradata Database

The DataDirect Connect XE *for* ODBC and DataDirect Connect64 XE *for* ODBC driver for the Teradata database support Teradata database servers when using the appropriate client software.

For the latest support information, visit the DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf.

The driver is supported in the Windows, UNIX, and Linux environments. See Environment-Specific Information on page 38 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the driver.

Driver Requirements

The driver requires Teradata Tools and Utilities (TTU) 8.2 or higher, which includes CLIv2, TGSS, and ICU client software, on all platforms. It requires TTU 12.0 to support 12.0 functionality.

Note: TTU 12.0 is not available for the Itanium II platform. You can use TTU 8.2 on an Itanium II client to connect to a Teradata 12.0 database, but functionality is limited to that of TTU 8.2.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Quick Start Connect on page 23 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See Using a Connection String on page 423 and Connection Option Descriptions on page 429 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX/Linux odbc.ini File

UNIX® On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See Environment Configuration on page 27 for basic setup information and Environment Variables on page 88 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). You can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

Connection Option Descriptions on page 429 lists driver connection string attributes that must be used in the odbc.ini file to set the value of the attributes. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

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 on page 429 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 Teradata is:

```
DSN=Teradata Tables;AS=User2;EnableDataEncryption=Yes
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Teradata.dsn;AS=User2;EnableDataEncryption=Yes
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER={DataDirect 7.1 Teradata};DBCSN=123.456.78.90;UIS=YES
```

Data Source Configuration through a GUI (Teradata)


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.

UNIX® On UNIX and Linux, data sources are stored in the odbc.ini file. See Data Source Configuration Through the System Information (odbc.ini) File on page 90 for detailed information about the specific steps necessary to configure a data source.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Teradata data source:

1. Start the ODBC Administrator by selecting its icon from the Progress DataDirect 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.

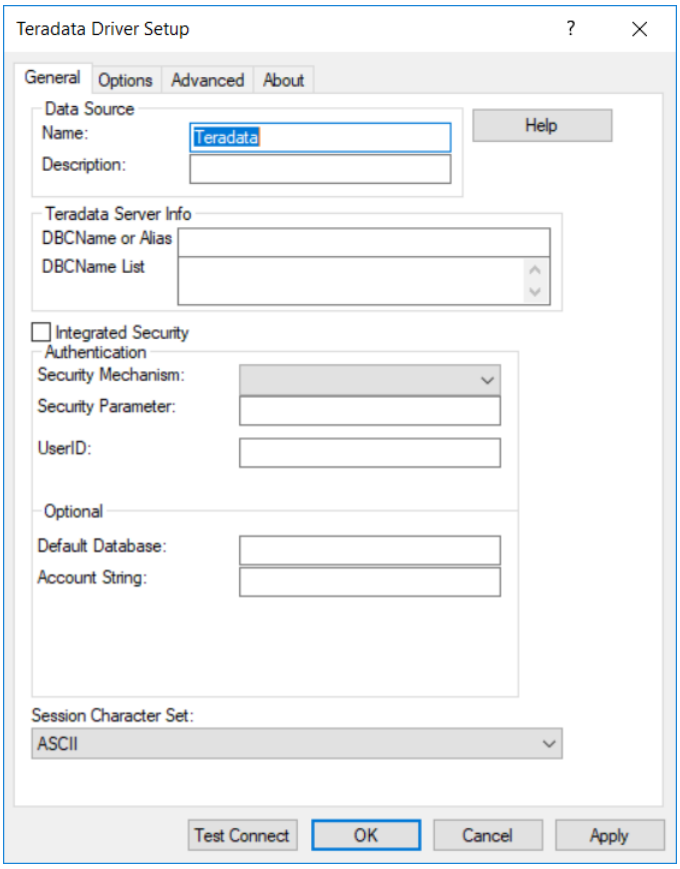
     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** 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.

The General tab of the Setup dialog box appears by default.

Figure 39: General tab



Note: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

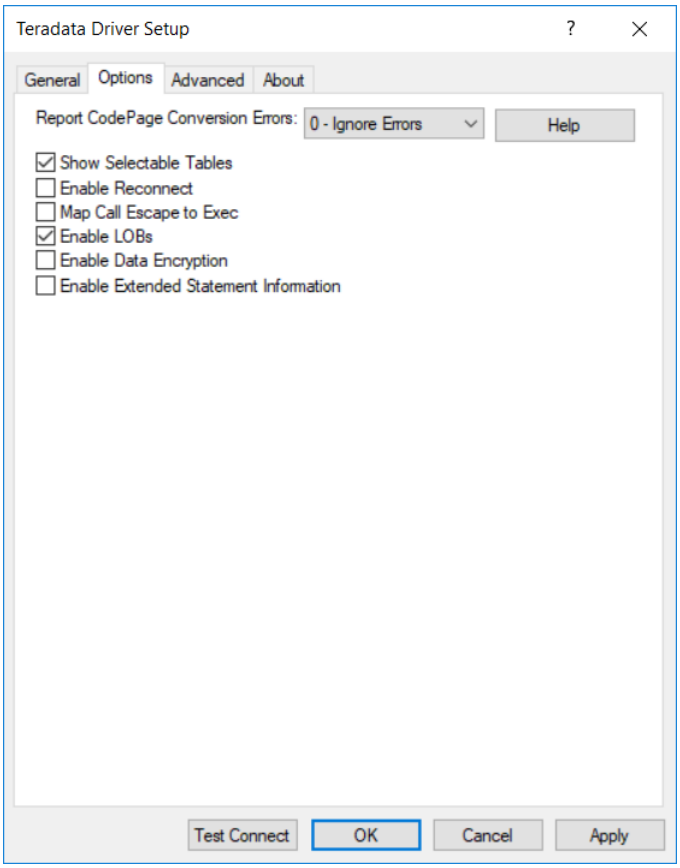
3. On this tab, provide values for the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: General	Default
Name on page 440	None
Description on page 435	None
DBCName or Alias on page 433	None
DBCName List on page 432	None
Integrated Security on page 438	Disabled
Security Mechanism on page 445	None
Security Parameter on page 446	None
UserID on page 448	None

Connection Options: General	Default
Default Database on page 434	None
Account String on page 430	None
Session Character Set on page 446	ASCII

4. Click the **Options** tab to specify additional configuration options.

Figure 40: Options tab



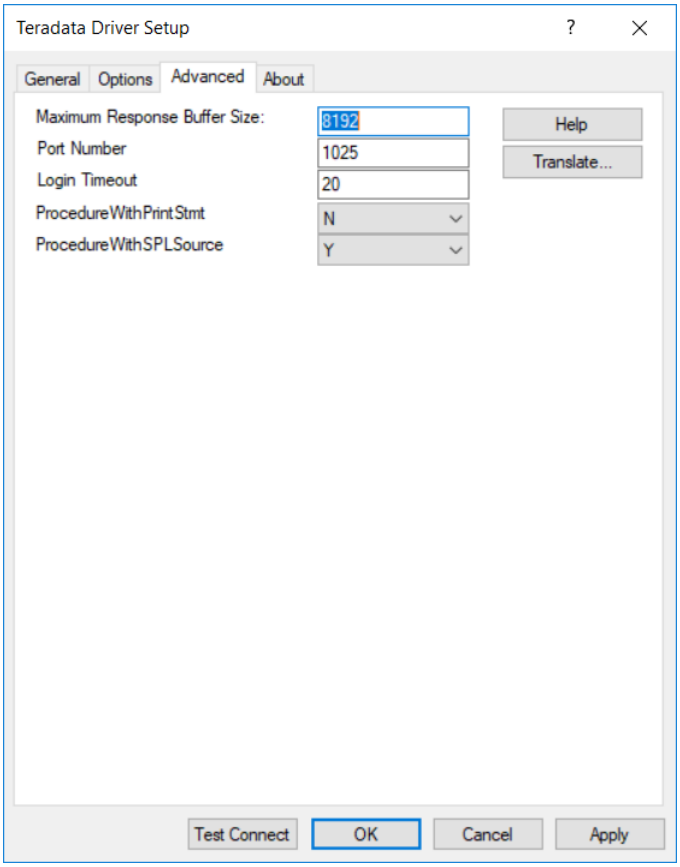
On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Options	Default
Report Codepage Conversion Errors on page 444	0 - Ignore Errors
Show Selectable Tables on page 447	Enabled
Enable Reconnect on page 437	Disabled
Map Call Escape To Exec on page 439	Disabled
Enable LOBs on page 436	Enabled

Connection Options: Options	Default
Enable Data Encryption on page 435	Disabled
Enable Extended Statement Information on page 436	Disabled

5. Optionally, click the **Advanced** tab to specify additional data source settings.

Figure 41: Advanced tab



On this tab, provide values for any of the options in the following table; then, click **Apply**. The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Maximum Response Buffer Size on page 440	8192
Port Number on page 441	1025
Login Timeout on page 439	20
ProcedureWithPrintStmt on page 442	N (No)
ProcedureWithSPLSource on page 442	Y (Yes)
IANAAppCodePage on page 437 UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. Progress DataDirect provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

6. 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 (Teradata) on page 428 for details). The information you enter in the logon dialog box during a test connect is not saved.

   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message.

     Verify that all required client software is properly installed. If it is not, you will see the message:

     Specified driver could not be loaded due to system error [xxx].

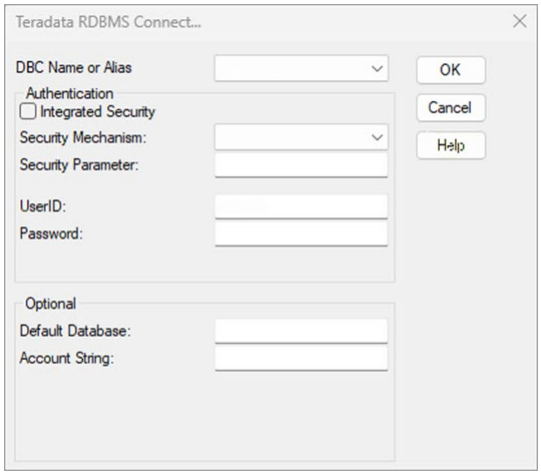
     Click **OK**.

7. Click **OK** or **Cancel**. If 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.

Using a Logon Dialog Box (Teradata)

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.

Figure 42: Teradata RDBMS Connect dialog box



In this dialog box, provide the following information:

1. Select an alias name or IP address of a Teradata server from the DBC Name or Alias drop-down list. The choices for this list are determined by the entries in DBC Name or Alias and DBCName List on the General tab of the driver Setup dialog box.
2. Select the Integrated Security check box to enable the user to connect to the database through Single Sign On (SSO) using one of the authentication mechanisms that support SSO. In this case, User Name, Password, and Domain are not required and are not available fields.

3. If you do not use Integrated Security, select a value from the Security Mechanism drop-down list to specify the authentication mechanism used for connections to the data source.
4. Type a string of characters in the Security Parameter field that is to be regarded as a parameter to the authentication mechanism. The string is ignored by the ODBC driver and is passed on to the TeraSSO function that is called to set the authentication mechanism.

   The characters [] {} () , ; ? * = ! @ must be enclosed in curly braces.
5. Other options that are displayed on the Logon Dialog box depend on the authentication mechanism selected. See the descriptions of these options under Security Mechanism on page 445.
6. Type the domain name for Third Party Sign On along with the username and password. If a domain name is not provided, then the local domain is assumed.
7. Type a default Teradata database (optional).
8. Type an account string to be used during the creation of a user account in the Teradata Database instead of providing account information during configuration of ODBC (optional).
9. Click **OK** to complete the logon and to update these values in the Registry.

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, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

The following table lists the connection string attributes supported by the driver.

Table 45: Teradata Attribute Names

Attribute (Short Name)	Default
AccountString (AS)	None
AuthenticationDomain (AD)	None
AuthenticationPassword (AP)	None
AuthenticationUserId (AUI)	None
CharacterSet (CS)	ASCII
Database (DB)	None
DataSourceName (DSN)	None
DBCName (DBCN)	None
DBCName List	None

Attribute (Short Name)	Default
Description (n/a)	None
EnableDataEncryption (EDE)	No (Disabled)
EnableExtendedStmtInfo (EESI)	No (Disabled)
EnableLOBs (EL)	Yes (Enabled)
EnableReconnect (ER)	No (Disabled)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntegratedSecurity (IS)	No (Disabled)
LoginTimeout (LTO)	20
MapCallEscapeToExec (MCETE)	No (Disabled)
MaxRespSize (MRS)	8192
Password (PWD)	None
PortNumber (PORT)	1025
PrintOption (PO)	N (No)
ProcedureWithSPLSource (PWSS)	Y (Yes)
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
SecurityMechanism (SECM)	None
SecurityParameter (SP)	None
ShowSelectableTables (SST)	Yes (Enabled)
TDProfile (TDP)	None
TDRole (TDR)	None
TDUserName (TDUN)	None
UserID (UID)	None

Account String

Attribute

AccountString (AS)

Purpose

An account string. For a complete description of account strings, refer to the *Teradata Database Administration Guide*.

Valid Values

string

where:

string

is an account string.

Default

None

GUI Tab

General tab

Authentication Password

Attribute

AuthenticationPassword (AP)

Purpose

The password for the Kerberos, LDAP, NTLM, and TD authentication mechanisms. The Authentication Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Authentication UserId

Attribute

AuthenticationUserId (AUI)

Purpose

The user ID for the Kerberos, LDAP, NTLM, and TD authentication mechanisms.

Valid Values

userid

where:

userid

is a valid user ID.

Default

None

GUI Tab

General tab

DBCName List

Attribute

DBCName List

Attribute

n/a

Purpose

A list of IP addresses or aliases to appear in the drop-down list of the Logon dialog box (see Using a Logon Dialog Box (Teradata) on page 428 for a description). The DBCName List option is not used as a runtime connection attribute.

Valid Values

ip_address | *alias* [,*ip_address* | *alias*][...]

where:

ip_address

is an IP address to appear in the drop-down list of the Logon dialog box.

alias

is an alias to appear in the drop-down list of the Logon dialog box.

Separate multiple IP addresses or aliases with commas. The same restrictions apply as described for the DBCName or Alias option.

Default

None

GUI Tab

General tab

DBCName or Alias

Attribute

DBCName (DBCN)

Purpose

The IP address or alias of the Teradata server.

Valid Values

IP_address | *alias*

where:

IP_address

is the IP address of the Teradata server.

alias

is the alias of the Teradata server.

Behavior

If set to *IP_address*, the time the driver waits for connections to be established is faster. The disadvantage is that if the server designated by that IP address is unavailable, the connection fails and the driver does not attempt to fail over to another IP address.

If set to *alias*, the time the driver waits for connections to be established is slower because the driver must search a local hosts file to resolve the alias to an IP address. The advantage is that the driver fails over the connection to an alternate IP address if the first address fails.

To use aliases, a local hosts file that maps aliases to IP addresses is required. Aliases cannot be more than eight characters. In the hosts file, you must specify the aliases and map each of them to an IP address in the order that you want the driver to attempt the connections. For example:

```
167.56.78.1 (NCR5100COP1)
167.56.78.2 (NCR5100COP2)
167.56.78.3 (NCR5100COP3)
```

where NCR5100 is an alias and COP*n* (where *n* = 1, 2, 3, ..., 128) is a suffix that sets the order of failover connection attempts. The eight-character limit on the alias does not include the suffix. You can enter a maximum of 128 COP (communications processor) entries per host.

Notes

- Although you must add a COP suffix to the alias in the hosts file, do *not* specify the suffix when entering the alias in the DBCName or Alias field of the Setup dialog box. Only specify the alias.

Default

None

GUI Tab

General tab

Default Database

Attribute

Database (DB)

Purpose

Specifies the name of the database to which you want to connect.

Valid Values

database_directory

where:

database_directory

is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

Default

None

GUI Tab

General tab

Default Role

Attribute

TDRole (TDR)

Purpose

Specifies the Teradata role for the LDAP authentication mechanism.

Valid Values

string

where:

string

is a valid role.

Default

None

GUI Tab

General tab

Description

Attribute

Description (n/a)

Purpose

Specifies an optional long description of a data source. This description is not used as a runtime connection attribute, but does appear in the ODBC.INI section of the Registry and in the odbc.ini file.

Valid Values

string

where:

string

is a description of a data source.

Default

None

GUI Tab

General tab

Enable Data Encryption

Attribute

EnableDataEncryption (EDE)

Purpose

Determines whether the driver uses data encryption.

Valid Values

1 | 0

Behavior

If set to 1 (Enabled), the driver encrypts data and communicates with the Teradata gateway using encryption.

If set to 0 (Disabled), the driver does not encrypt data except for logon information.

Notes

- Before you use this value, verify that the server is encryption capable. Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) required to encrypt and decrypt data.

Default

0 (Disabled)

GUI Tab

Options tab

Enable Extended Statement Information

Attribute

EnableExtendedStmtInfo (EESI)

Purpose

Determines whether the driver supports extended statement information.

Valid Values

1 | 0

Behavior

If set to 1 (Enabled), the driver queries the server to see if it supports the Statement Information parcel. If the server supports the Statement Information parcel, the driver requests the Statement Information parcel and enables auto-generated key retrieval and SQLDescribeParam support. Use this value if you want to enable the Return Generated Keys option.

If set to 0 (Disabled), the driver does not attempt to expose auto-generated key retrieval or SQLDescribeParam.

Default

0 (Disabled)

GUI Tab

Options tab

Enable LOBs

Attribute

EnableLOBs (EL)

Purpose

Determines whether the driver enforces native LOB data type mapping.

Valid Values

1 | 0

Behavior

If set to 1 (Enabled), the driver enforces native LOB data type mapping as described:

- ODBC data type SQL_LONGVARBINARY is mapped to the Teradata BLOB feature.

- ODBC data type SQL_LONGVARCHAR is mapped to the Teradata CLOB feature.

If set to 0 (Disabled), the driver provides backward compatibility for applications without LOB support that are using a version of Teradata Database prior to V2R5.1. The mappings are:

- ODBC data type SQL_LONGVARBINARY is mapped to the Teradata VARBYTE(32000) feature.
- ODBC data type SQL_LONGVARCHAR is mapped to the Teradata LONG VARCHAR feature.

This value can improve performance if your application does not send data to, or retrieve it from, LOB columns. You may receive an error if you disable this option and try to retrieve data from a LOB column.

Default

1 (Enabled)

GUI Tab

Options tab

Enable Reconnect

Attribute

EnableReconnect (ER)

Purpose

Determines whether the driver will reconnect after a system crash or reset is detected.

Valid Values

1 | 0

Behavior

If set to 1 (Enabled), the driver attempts to reconnect to the saved sessions; however, sessions cannot be reconnected until the Teradata system is available. After a session has been reconnected, applications can expect to receive error messages describing why the ODBC function failed, as well as a status report describing the post-recovery state.

If set to 0 (Disabled), the driver does not attempt to reconnect to the saved sessions.

Default

0 (Disabled)

GUI Tab

Options tab

IANAAppCodePage

Attribute

IANAAppCodePage (IACP)

Purpose

An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled or if your database character set is not Unicode. The value you specify must match the database character encoding and the system locale.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values

IANA_code_page

where:

IANA_code_page

is one of the valid values listed in "IANAAppCodePage values" in the *Progress DataDirect for ODBC Drivers Reference*. The value must match the database character encoding and the system locale.

Notes

Refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for details.

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

Advanced tab

Integrated Security

Attribute

IntegratedSecurity (IS)

Purpose

Determines whether the driver allows the user to connect to the database using Single Sign On (SSO) through an authentication mechanism that supports SSO.

Valid Values

Yes | No

Behavior

If set to Yes (Enabled), SSO is allowed. The driver uses the operating system user ID and password.

If set to No (Disabled), you must specify a value for the UserID option.

Default

No (Disabled)

GUI Tab

General tab

Login Timeout

Attribute

LoginTimeout (LTO)

Purpose

The number of seconds to wait when establishing a virtual circuit with Teradata for login.

Valid Values

x

where:

x

is a positive integer.

Behavior

If set to *x*, the driver waits the specified number of seconds.

Default

20

GUI Tab

Advanced tab

Map Call Escape To Exec

Attribute

MapCallEscapeToExec (MCETE)

Purpose

Determines whether the driver converts the `{CALL <name>(...)}` statement to `EXEC name(...)`.

Valid Values

Yes | No

Behavior

If set to Yes (Enabled), the driver considers the {CALL <name>(...)} statement as the SQL for MACRO execution and converts it to EXEC *name*(...).

If set to No (Disabled), the driver does not convert {CALL *name*(...)} statements to EXEC *name*(...), and considers them as CALL statements for Stored Procedure Execution.

Default

No (Disabled)

GUI Tab

Options tab

Maximum Response Buffer Size

Attribute

MaxRespSize (MRS)

Purpose

The size of the Teradata response buffer used for SQL requests. This value may be adjusted dynamically if Teradata cannot send a result within the defined size.

Valid Values

A positive integer from 1 to 65477

Behavior

If using a slow TCP/IP interface, such as PPP or SLIP, enter a smaller value. If you expect to retrieve large result sets in a LAN environment, set a larger value.

Default

8192

GUI Tab

Advanced tab

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

Password

Attribute

Password (PWD)

Purpose

The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values

pwd

where:

pwd

is a valid password.

Default

None

GUI Tab

n/a

Port Number

Attribute

PortNumber (PORT)

Purpose

The port number of the server listener.

Valid Values

port_name

where:

port_name

is the port number of the server listener. Check with your database administrator for the correct number.

Default

1025

GUI Tab

Advanced tab

ProcedureWithPrintStmt

Attribute

PrintOption (PO)

Purpose

Determines whether the driver activates the print option when creating stored procedures.

Valid Values

P | N

Behavior

If set to P (Print), the driver activates the print option.

If set to N (No), the driver does not activate the print option.

Default

N (No)

GUI Tab

Advanced tab

ProcedureWithSPLSource

Attribute

ProcedureWithSPLSource (PWSS)

Purpose

Determines whether the drive specifies SPL text when creating stored procedures.

Valid Values

Y | N

Behavior

If set to Y (Yes), the driver specifies SPL text.

If set to N (No), the driver does not specify SPL text.

Default

Y (Yes)

GUI Tab

Advanced tab

Profile

Attribute

TDProfile (TDP)

Purpose

Specifies the Teradata profile for the LDAP authentication mechanism.

Valid Values

string

where:

string

is a valid profile.

Notes

This connection option should be used only if the directory maps the Authentication UserID to multiple database users or profiles. The value supplied must be one of the mapped values.

Default

None

GUI Tab

General tab

Realm

Attribute

AuthenticationDomain (AD)

Purpose

Specifies the domain appropriate to the selected authentication mechanism

Valid Values

string

where:

string

is a valid domain.

Default

None

GUI Tab

None

Report Codepage Conversion Errors

Attribute

ReportCodepageConversionErrors (RCCE)

Purpose

Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another.

An error message or warning can occur if an ODBC call causes a conversion error, or if an error occurs during code page conversions to and from the database or to and from the application. The error or warning generated is `Code page conversion error encountered`. In the case of parameter data conversion errors, the driver adds the following sentence: `Error in parameter` *x*, where *x* is the parameter number. The standard rules for returning specific row and column errors for bulk operations apply.

Valid Values

0 | 1 | 2

Behavior

If set to 0 (Ignore Errors), the driver substitutes 0x1A for each character that cannot be converted and does not return a warning or error.

If set to 1 (Return Error), the driver returns an error instead of substituting 0x1A for unconverted characters.

If set to 2 (Return Warning), the driver substitutes 0x1A for each character that cannot be converted and returns a warning.

Default

0 (Ignore Errors)

GUI Tab

Options tab

Security Mechanism

Attribute

SecurityMechanism (SECM)

Purpose

The authentication method to be used by the driver for connections to the database.

Valid Values

TD1 | TD2 | ldap | KRB5 | KRB5C | NTLM | NTLMC

Behavior

If set to TD1, the driver uses Teradata 1.

If set to TD2, the driver uses Teradata 2.

If set to ldap, the driver uses LDAP.

If set to KRB5, the driver uses Kerberos on Windows clients working with Windows servers if the server is V2R6.0.

If set to KRB5C, the driver uses Kerberos Compatibility on Windows clients working with Windows servers if the server is pre-V2R6.0.

If set to NTLM, the driver uses NTLM on Windows clients working with Windows servers if the server is V2R6.0.

If set to NTLMC, the driver uses NTLM Compatibility on Windows clients working with Windows servers if the server is pre-V2R6.0.

Note: Kerberos and NTLM are enabled through the Teradata client. See your Teradata documentation for requirements.

The following options may appear, based on the selected method:

- *No mechanism selected*
  - **User name**: A user name for the default Teradata database. If TeraSSO allows fully qualified user names, the user name may contain a domain or realm, for example, {judy@linedata}. Values containing a character like @ must be enclosed in braces.
- *KRB5 and KRB5C*
  - **Authentication UserID**: The Kerberos user ID.
  - **Realm**: The Kerberos domain. (The equivalent connection string attribute is AuthenticationDomain.)
- *LDAP*
  - **Authentication UserID**: The LDAP user ID.
  - **Realm**: The LDAP domain. (The equivalent connection string attribute is AuthenticationDomain.)
  - **TD User name**: The Teradata user name.
  - **Profile**: The Teradata Profile. (The equivalent connection string attribute is TDProfile.)
  - **Default Role**: The Teradata Role. (The equivalent connection string attribute is TDRole.)

- *NTLM and NTLMC*
  - **Authentication UserID**: The NTLM user ID.
  - **Realm**: The NTLM domain. (The equivalent connection string attribute is AuthenticationDomain.)
- *TD1 and TD2*
  - **Authentication UserID**: The TD1 or TD2 user ID.

Other parameters for the authentication mechanism can be entered in the Security Parameter field.

Default

None

GUI Tab

General tab

Security Parameter

Attribute

SecurityParameter (SP)

Purpose

A string that is passed as a parameter to the authentication method. The string is ignored by the ODBC driver and is passed to the TeraSSO function that is called to set the authentication method.

Valid Values

`string`

where:

`string`

is a string of characters. The characters [] {} () , ; ? * = ! @ must be enclosed in curly braces.

Default

None

GUI Tab

General tab

Session Character Set

Attribute

CharacterSet (CS)

Purpose

A character set used to override the Teradata character set.

Valid Values

ASCII | UTF16 (valid only for V2R6.*x* servers) | LATIN1252_0A | LATIN9_0A | LATIN1_0A | Shift-JIS | EUC | BIG5 | GB | NetworkKorean

The specified character set must be installed on the database.

Default

ASCII

GUI Tab

General tab

Show Selectable Tables

Attribute

ShowSelectableTables (SST)

Purpose

Determines whether the driver supports X views.

Valid Values

Yes | No

Behavior

If set to Yes (Enabled), SQLTables() and SQLProcedures() use dbc.tablesX and dbc.databasesX instead of dbc.tables and dbc.databases. Also, SQLColumns() and SQLProcedureColumns() use dbc.columnsX instead of dbc.columns. SqlStatistics() uses dbc.tablesizeX instead of dbc.tablesize. The X tables only contain information that the user has permission to access. These tables are optional for Teradata, so verify that they exist.

If set to No (Disabled), SQLTables() and SQLProcedures() use dbc.tables and dbc.databases. Also, SQLColumns() and SQLProcedureColumns() use dbc.columns. SqlStatistics() uses dbc.tablesize.

Default

Yes (Enabled)

GUI Tab

Options tab

TDUserName

Attribute

TDUserName (TDUN)

Purpose

Specifies the Teradata user name for the LDAP authentication mechanism.

Valid Values

user_name

where:

user_name

is a valid user name.

Notes

This connection option should be used only if the directory maps the Authentication UserID to multiple database users or profiles. The value supplied must be one of the mapped values.

Default

None

GUI Tab

General tab

UserID

Attribute

UserID (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.

Behavior

The user name is interpreted in the context of the authentication mechanism. If, for example, the authentication mechanism is NTLM, the user name is assumed to be a Windows user name.

If TeraSSO allows fully qualified user names, the user name may contain a domain or realm, for example, {judy@linedata}. Values containing a character such as @ must be enclosed in curly braces.

SSO is indicated by the absence of a UserID.

Default

None

GUI Tab

General tab

Data Types

The following table shows how the Teradata data types map to the standard ODBC data types.

Table 46: Teradata Data Types

Teradata	ODBC
Blob[21]	SQL_LONGVARBINARY
Bigint	SQL_BIGINT
Byte	SQL_BIT
Byteint	SQL_TINYINT
Char	SQL_CHAR
Clob[22]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_FLOAT
Integer	SQL_INTEGER
Interval day	SQL_INTERVAL_DAY
Interval day to hour	SQL_INTERVAL_DAY_TO_HOUR
Interval day to minute	SQL_INTERVAL_DAY_TO_MINUTE
Interval day to second	SQL_INTERVAL_DAY_TO_SECOND
Interval hour	SQL_INTERVAL_HOUR
Interval hour to minute	SQL_INTERVAL_HOUR_TO_MINUTE
Interval hour to second	SQL_INTERVAL_HOUR_TO_SECOND

[21] If no LOB support, VARBYTE(32000).
[22] If no LOB support, LONGVARCHAR.

Teradata	ODBC
Interval minute	SQL_INTERVAL_MINUTE
Interval minute to second	SQL_INTERVAL_MINUTE_TO_SECOND
Interval month	SQL_INTERVAL_MONTH
Interval second	SQL_INTERVAL_SECOND
Interval year	SQL_INTERVAL_YEAR
Interval year to month	SQL_INTERVAL_YEAR_TO_MONTH
Number	SQL_DOUBLE
Number (p)	SQL_BIGINT | SQL_DECIMAL[23]
Number (p, s)	SQL_DECIMAL
Numeric	SQL_NUMERIC
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR

Unicode Support

The driver supports Unicode data types. The following table shows how the Teradata data types map to the Unicode data types, but only when CharacterSet is set to UTF-16.

Table 47: Teradata Unicode Data Types

Teradata	Unicode
char () charset Unicode	SQL_WCHAR
clob charset Unicode	SQL_WLONGVARCHAR
varchar () charset Unicode	SQL_WVARCHAR

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

[23] When precision is less than or equal to 19, Number maps to SQL_BIGINT. When precision is greater than 19, it maps to Decimal.

See UTF-16 Applications on UNIX and Linux on page 101 for related details.

Also, refer to "Internationalization, localization, and Unicode" in the *Progress DataDirect for ODBC Drivers Reference* for a more detailed explanation of Unicode.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See Persisting a Result Set as an XML Data File on page 48 for details about implementation.

Isolation and Lock Levels Supported

Teradata supports isolation levels 0 (read uncommitted) and 3 (serializable).

Refer to "Locking and isolation levels" in the *Progress DataDirect for ODBC Drivers Reference* for details.

SQL Support

The driver supports the minimum SQL grammar.

ODBC Conformance Level

The driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions.

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for a list of supported API functions.

Number of Connections and Statements Supported

The driver supports multiple connections and 16 statements per connection to the Teradata database system.

10

Supported SQL Statements and Extensions

This chapter includes information on supported SQL functionality for the Impala Wire Protocol and flat-file drivers.

For details, see the following topics:

- SQL Statements for Flat-File Drivers
- SQL Functionality for the Impala Wire Protocol Driver

SQL Statements for Flat-File Drivers

This chapter describes the SQL statements that you can use with the flat-file drivers (Btrieve, dBASE, and Text). Any exceptions to the supported SQL functionality described in this chapter are documented in the individual flat-file driver chapters in the *DataDirect Connect Series for ODBC User's Guide*.

The database drivers parse SQL statements and translate them into a form that the database can understand. The SQL statements described in this chapter let you:

- Read, insert, update, and delete rows from a database
- Create new tables
- Drop existing tables

These SQL statements allow your application to be portable across other databases.

Select Statement

The form of the Select statement supported by the flat-file drivers is:

```
SELECT [DISTINCT] {* | column_expression, ...}
FROM table_names [table_alias] ...
[ WHERE expr1rel_operatorexpr2 ]
[ GROUP BY {column_expression, ...} ]
[ HAVING expr1rel_operatorexpr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY {sort_expression [DESC | ASC]}, ... ]
[ FOR UPDATE [OF {column_expression, ...}] ]
```

Select Clause

Follow Select with a list of column expressions you want to retrieve or an asterisk (*) to retrieve all fields.

```
SELECT [DISTINCT] {* | column_expression, [[AS] column_alias]. . .}
```

column_expression can be simply a field name (for example, LAST_NAME). More complex expressions may include mathematical operations or string manipulation (for example, SALARY * 1.05). See SQL Expressions on page 457 for details.

column_alias can be used to give the column a descriptive name. For example, to assign the alias DEPARTMENT to the column DEP:

```
SELECT dep AS department FROM emp
```

Separate multiple column expressions with commas (for example, LAST_NAME, FIRST_NAME, HIRE_DATE).

Field names can be prefixed with the table name or alias. For example, EMP.LAST_NAME or E.LAST_NAME, where E is the alias for the table EMP.

The Distinct operator can precede the first column expression. This operator eliminates duplicate rows from the result of a query. For example:

```
SELECT DISTINCT dep FROM emp
```

Aggregate Functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of rows. An aggregate can be used with a field name (for example, AVG(SALARY)) or in combination with a more complex column expression (for example, AVG(SALARY * 1.07)). The column expression can be preceded by the Distinct operator. The Distinct operator eliminates duplicate values from an aggregate expression. For example:

```
COUNT (DISTINCT last_name)
```

In this example, only distinct last name values are counted.

The following table lists valid aggregate functions.

Table 48: Aggregate Functions

Aggregate	Returns
SUM	The total of the values in a numeric field expression. For example, SUM(SALARY) returns the sum of all salary field values.

AVG	The average of the values in a numeric field expression. For example, AVG(SALARY) returns the average of all salary field values.
COUNT	The number of values in any field expression. For example, COUNT(NAME) returns the number of name values. When using COUNT with a field name, COUNT returns the number of non-NULL field values. A special example is COUNT(*), which returns the number of rows in the set, including rows with NULL values.
MAX	The maximum value in any field expression. For example, MAX(SALARY) returns the maximum salary field value.
MIN	The minimum value in any field expression. For example, MIN(SALARY) returns the minimum salary field value.

From Clause

The From clause indicates the tables to be used in the Select statement. The format of the From clause is:

```
FROM table_names [table_alias]
```

table_names can be one or more simple table names in the current working directory or complete path names.

table_alias is a name used to refer to a table in the rest of the Select statement. Database field names may be prefixed by the table alias. Given the table specification:

```
FROM emp E
```

you may refer to the LAST_NAME field as E.LAST_NAME. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

If you are joining more than one table, you can use LEFT OUTER JOIN, which includes non-matching rows in the first table you name. For example:

```
SELECT * FROM T1 LEFT OUTER JOIN T2 on T1.key = T2.key
```

Where Clause

The Where clause specifies the conditions that rows must meet to be retrieved. The Where clause contains conditions in the form:

```
WHERE expr1rel_operatorexpr2
```

expr1 and *expr2* can be field names, constant values, or expressions.

rel_operator is the relational operator that links the two expressions. See SQL Expressions on page 457 for details.

For example, the following Select statement retrieves the names of employees that make at least $20,000.

```
SELECT last_name,first_name FROM emp WHERE salary >= 20000
```

Group By Clause

The Group By clause specifies the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values. It has the following form:

```
GROUP BY column_expressions
```

column_expressions must match the column expression used in the Select clause. A column expression can be one or more field names of the database table, separated by a comma (,) or one or more expressions, separated by a comma (,). See SQL Expressions on page 457 for details.

The following example sums the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

Having Clause

The Having clause enables you to specify conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a Group By clause. It has the following form:

```
HAVING expr1rel_operatorexpr2
```

expr1 and *expr2* can be field names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause.

rel_operator is the relational operator that links the two expressions. See SQL Expressions on page 457 for details.

The following example returns only the departments whose sums of salaries are greater than $200,000:

```
SELECT dept_id, sum(salary) FROM emp
GROUP BY dept_id HAVING sum(salary) > 200000
```

Union Operator

The Union operator combines the results of two Select statements into a single result. The single result is all the returned rows from both Select statements. By default, duplicate rows are not returned. To return duplicate rows, use the All keyword (UNION ALL). The form is:

```
SELECT statement
UNION ALL
SELECT statement
```

When using the Union operator, the Select lists for each Select statement must have the same number of column expressions with the same data types, and must be specified in the same order. For example:

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

This example has the same number of column expressions, and each column expression, in order, has the same data type.

The following example is *not* valid because the data types of the column expressions are different (salary from emp has a different data type than last_name from raises). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

Order By Clause

The Order By clause indicates how the rows are to be sorted. The form is:

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

sort_expression can be field names, expressions, or the positioned number of the column expression to use.

The default is to perform an ascending (ASC) sort.

For example, to sort by last_name and then by first_name, you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, last_name is the second column expression following Select, so Order By 2 sorts by last_name.

For Update Clause

The For Update clause locks the rows of the database table selected by the Select statement. The form is:

```
FOR UPDATE OF column_expressions
```

column_expressions is a list of field names in the database table that you intend to update, separated by a comma (,).

The following example returns all rows in the employee database that have a salary field value of more than $20,000. When each record is fetched, it is locked. If the record is updated or deleted, the lock is held until you commit the change. Otherwise, the lock is released when you fetch the next record.

```
SELECT * FROM emp WHERE salary > 20000   FOR UPDATE OF last_name, first_name,
salary
```

SQL Expressions

Expressions are used in the Where clauses, Having clauses, and Order By clauses of SQL Select statements.

Expressions enable you to use mathematical operations as well as character string and date manipulation operators to form complex database queries.

The most common expression is a simple field name. You can combine a field name with other expression elements.

Valid expression elements are as follows:

- Field Names
- Constants
- Exponential notation
- Numeric operators

- Character operators
- Date operators
- Relational operators
- Logical operators
- Functions

Constants

Constants are values that do not change. For example, in the expression PRICE * 1.05, the value 1.05 is a constant.

You must enclose character constants in pairs of single (') or double (") quotation marks. To include a single quotation mark in a character constant enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t'). Similarly, if the constant is enclosed by double quotation marks, use two double quotation marks to include one.

You must enclose date and time constants in braces ({}), for example, {01/30/89} and {12:35:10}. The form for date constants is MM/DD/YY or MM/DD/YYYY. The form for time constants is HH:MM:SS.

The logical constants are .T. and 1 for True and .F. and 0 for False. For portability, use 1 and 0.

Exponential Notation

You can include exponential notation in expression elements. For example:

```
SELECT col1, 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * col2
```

Numeric Operators

You can include the following operators in numeric expressions:

Operator	Meaning
+	Addition
–	Subtraction
*	Multiplication
/	Division
**	Exponentiation
^	Exponentiation

The following table shows examples of numeric expressions. For these examples, assume salary is 20000.

Example	Resulting value
`salary + 10000`	`30000`
`salary * 1.1`	`22000`
`2 ** 3`	`8`

You can precede numeric expressions with a unary plus (+) or minus (–). For example, `–(salary * 1.1)` is -22000.

Character Operators

Character expressions can include the following operators:

Operator	Meaning
+	Concatenation, keeping trailing blanks.
–	Concatenation, moving trailing blanks to the end.

The following table shows examples of character expressions. In the examples, last_name is `'JONES  '` and first_name is `'ROBERT  '`.

Example	Resulting Value
first_name + last_name	'ROBERT JONES '
first_name – last_name	'ROBERTJONES '

Note: Some flat-file drivers return character data with trailing blanks as shown in the table; however, you cannot rely on the driver to return blanks. If you want an expression that works regardless of whether the drivers return trailing blanks, use the TRIM function before concatenating strings to make the expression portable. For example:

```
TRIM(first_name) + '' + TRIM(last_name)
```

Date Operators

You can include the following operators in date expressions:

Operator	Meaning
+	Add a number of days to a date to produce a new date.
–	The number of days between two dates, or subtract a number of days from a date to produce a new date.

The following table shows examples of date expressions. In these examples, hire_date is {01/30/1990}.

Example	Resulting Value
`hire_date + 5`	`{02/04/1990}`
`hire_date – {01/01/1990}`	`29`
`hire_date – 10`	`{01/20/1990}`

Relational Operators

Relational operators separating any two expressions can be any one of those listed in the following table.

Table 49: Relational Operators

Operator	Meaning
=	Equal.
<>	Not Equal.

Operator	Meaning
>	Greater Than.
>=	Greater Than or Equal.
<	Less Than.
<=	Less Than or Equal.
Like	Matching a pattern.
Not Like	Not matching a pattern.
Is NULL	Equal to NULL.
Is Not NULL	Not Equal to NULL.
Between	Range of values between a lower and upper bound.
In	A member of a set of specified values or a member of a subquery.
Exists	True if a subquery returned at least one record.
Any	Compares a value to each value returned by a subquery. Any must be prefaced by =, <>, >, >=, <, or <=.=Any is equivalent to In.
All	Compares a value to each value returned by a subquery. All must be prefaced by =, <>, >, >=, <, or <=.

The following list shows some examples of relational operators:

```
salary <= 40000
dept = 'D101'
hire_date > {01/30/1989}
salary + commission >= 50000
last_name LIKE 'Jo%'
salary IS NULL
salary BETWEEN 10000 AND 20000
WHERE salary = ANY (SELECT salary FROM emp WHERE dept = 'D101')
WHERE salary > ALL (SELECT salary FROM emp WHERE dept = 'D101')
```

Logical Operators

Two or more conditions may be combined to form more complex criteria. When two or more conditions are present, they must be related by AND or OR. For example:

```
salary = 40000 AND exempt = 1
```

The logical NOT operator is used to reverse the meaning. For example:

```
NOT (salary = 40000 AND exempt = 1)
```

Operator Precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. The following table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.

Table 50: Operator Precedence

Precedence	Operator
1	Unary -, Unary +
2	**
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, LIKE, NOT LIKE, IS NULL, IS NOT NULL, BETWEEN, IN, EXISTS, ANY, ALL
6	NOT
7	AND
8	OR

The following example shows the importance of precedence:

```
WHERE salary > 40000 OR
hire_date > {01/30/1989} AND
dept = 'D101'
```

Because AND is evaluated first, this query retrieves employees in department D101 hired after January 30, 1989, as well as every employee making more than $40,000, no matter what department or hire date.

To force the clause to be evaluated in a different order, use parentheses to enclose the conditions to be evaluated first. For example:

```
WHERE (salary > 40000 OR hire_date > {01/30/1989})
AND dept = 'D101'
```

retrieves employees in department D101 that either make more than $40,000 or were hired after January 30, 1989.

Functions

The flat-file drivers support a number of functions that you may use in expressions. In the following tables, the functions are grouped according to the type of result they return.

Table 51: Functions that Return Character Strings

Function	Description
CHR	Converts an ASCII code into a one-character string. `CHR(67)` returns `C`.
RTRIM	Removes trailing blanks from a string. `RTRIM('ABC  ')` returns `ABC`.
TRIM	Removes trailing blanks from a string. `TRIM('ABC  ')` returns `ABC`.
LTRIM	Removes leading blanks from a string. `LTRIM('  ABC')` returns `ABC`.
UPPER	Changes each letter of a string to uppercase. `UPPER('Allen')` returns `ALLEN`.
LOWER	Changes each letter of a string to lowercase. `LOWER('Allen')` returns `allen`.
LEFT	Returns leftmost characters of a string. `LEFT('Mattson',3)` returns `Mat`.
RIGHT	Returns rightmost characters of a string. `RIGHT('Mattson',4)` returns `tson`.
SUBSTR	Returns a substring of a string. Parameters are the string, the first character to extract, and the number of characters to extract (optional). `SUBSTR('Conrad',2,3)` returns `onr`. `SUBSTR('Conrad',2)` returns `onrad`.
SPACE	Generates a string of blanks. `SPACE(5)` returns `'  '`.

Function	Description
DTOC	Converts a date to a character string. An optional second parameter determines the format of the result: 0 (the default) returns MM/DD/YY. 1 returns DD/MM/YY. 2 returns YY/MM/DD. 10 returns MM/DD/YYYY. 11 returns DD/MM/YYYY. 12 returns YYYY/MM/DD. An optional third parameter specifies the date separator character. If not specified, a slash (/) is used. `DTOC({01/30/1997})` returns `01/30/97`. `DTOC({01/30/1997}, 0)` returns `01/30/97`. `DTOC({01/30/1997}, 1)` returns `30/01/97`. `DTOC({01/30/1997}, 2,'-')` returns `97-01-30`.
DTOS	Converts a date to a character string using the format YYYYMMDD. `DTOS({01/23/1990})` returns `19900123`.
IIF	Returns one of two values, true or false. Parameters are a logical expression, the true value, and the false value. If the logical expression evaluates to true, the function returns the true value. Otherwise, it returns the false value. `IIF(salary>20000,'BIG','SMALL')` returns `BIG` if `salary` is greater than 20000. If not, it returns `SMALL`.
STR	Converts a number to a character string. Parameters are the number, the total number of output characters (including the decimal point), and optionally the number of digits to the right of the decimal point. `STR(12.34567,4)` returns `12`. `STR(12.34567,4,1)` returns `12.3`. `STR(12.34567,6,3)` returns `12.346`.
STRVAL	Converts a value of any type to a character string. `STRVAL('Woltman')` returns `Woltman`. `STRVAL({12/25/1953})` returns `12/25/1953`. `STRVAL (5 * 3)` returns `15`. `STRVAL (4 = 5)` returns `'False'`.
TIME	Returns the time of day as a character string.At 9:49 PM, `TIME()` returns `21:49:00`.

Function	Description
TTOC	**Note:** This function applies only to flat-file drivers that support SQL_TIMESTAMP: the Btrieve driver and the dBASE (access to FoxPro 3.0) driver. Converts a timestamp to a character string. An optional second parameter determines the format of the result: When set to 0 or none (the default), MM/DD/YY HH:MM:SS AM is returned. When set to 1, YYYYMMDDHHMMSS is returned, which is a suitable format for indexing. `TTOC({1992-04-02 03:27:41})` returns 04/02/92 03:27:41 AM. `TTOC({1992-04-02 03:27:41, 1})` returns 19920402032741
USERNAME	For Btrieve, the logon ID specified at connect time is returned. For all other flat-file drivers, an empty string is returned.

Table 52: Functions that Return Numbers

Function	Description
MOD	Divides two numbers and returns the remainder of the division. `MOD(10,3)` returns `1`.
LEN	Returns the length of a string. `LEN('ABC')` returns `3`.
MONTH	Returns the month part of a date. `MONTH({01/30/1989})` returns `1`.
DAY	Returns the day part of a date. `DAY({01/30/1989})` returns `30`.
YEAR	Returns the year part of a date. `YEAR({01/30/1989})` returns `1989`.
MAX	Returns the larger of two numbers. `MAX(66,89)` returns `89`.
DAYOFWEEK	Returns the day of week (1-7) of a date expression. `DAYOFWEEK({05/01/1995})` returns `5`.
MIN	Returns the smaller of two numbers. `MIN(66,89)` returns `66`.

Function	Description
POW	Raises a number to a power. `POW(7,2)` returns `49`.
INT	Returns the integer part of a number `.INT(6.4321)` returns `6`.
ROUND	Rounds a number. `ROUND(123.456, 0)` returns `123`. `ROUND(123.456, 2)` returns `123.46`. `ROUND(123.456, –2)` returns `100`.
NUMVAL	Converts a character string to a number. If the character string is not a valid number, a zero (0) is returned. `NUMVAL('123')` returns the number `123`.
VAL	Converts a character string to a number. If the character string is not a valid number, a zero (0) is returned. `VAL('123')` returns the number `123`.

Table 53: Functions that Return Dates

Function	Description
DATE	Returns today’s date. If today is 12/25/1999, `DATE()` returns `{12/25/1999}`.
TODAY	Returns today's date. If today is 12/25/1999, `TODAY()` returns `{12/25/1999}`.
DATEVAL	Converts a character string to a date. `DATEVAL('01/30/1989')` returns `{01/30/1989}`.
CTOD	Converts a character string to a date. An optional second parameter specifies the format of the character string: 0 (the default) returns MM/DD/YY, 1 returns DD/MM/YY, and 2 returns YY/MM/DD. `CTOD('01/30/1989')` returns `{01/30/1989}`. `CTOD('01/30/1989',1)` returns `{30/01/1989}`.

The following examples use some of the number and date functions.

Retrieve all employees that have been with the company at least 90 days:

```
SELECT first_name, last_name FROM emp
```

```
WHERE DATE() – hire_date >= 90
```

Retrieve all employees hired in January of this year or last year:

```
SELECT first_name, last_name FROM emp
  WHERE MONTH(hire_date) = 1
  AND (YEAR(hire_date) = YEAR(DATE())
  OR YEAR(hire_date) = YEAR(DATE()) – 1)
```

Create and Drop Table Statements

The flat-file drivers support SQL statements to create and delete database files. The Create Table statement is used to create files and the Drop Table statement is used to delete files.

Create Table

The form of the Create Table statement is:

CREATE TABLE *table_name* (*col_definition*[,*col_definition*,...])

table_name can be a simple table name or a full path name. A table name is preferred for portability to other SQL data sources. If a table name is used, the file is created in the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is created in the directory specified as the database directory in .odbc.ini. If you did not specify a database directory in either place, the file is created in the current working directory at connect time.

col_definition is the column name, followed by the data type, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is "not NULL." Not all flat-file tables support "not NULL" columns. In the cases where "not NULL" is not supported, this restriction is ignored and the driver returns a warning if "not NULL" is specified for a column. The "not NULL" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not NULL" restriction).

A sample Create Table statement to create an employee database table is:

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL,
  first_name CHAR(12) NOT NULL,
  salary NUMERIC (10,2) NOT NULL,
  hire_date DATE NOT NULL)
```

Drop Table

The form of the Drop Table statement is:

DROP TABLE *table_name*

table_name can be a simple table name (emp) or a full path name. A table name is preferred for portability to other SQL data sources. If a table name is used, the file is dropped from the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is deleted from the directory specified as the database directory in .odbc.ini. If you did not specify a database directory in either of these places, the file is dropped from the current working directory at connect time.

A sample Drop Table statement to delete the emp table is:

```
DROP TABLE emp
```

Insert Statement

The Insert statement is used to add new rows to a database table. With it, you can specify either of the following options:

- A list of values to be inserted as a new record
- A Select statement that copies data from another table to be inserted as a set of new rows

The form of the Insert statement is:

```
INSERT INTO table_name [(col_name, ...)]
{VALUES (expr, ...) | select_statement}
```

table_name can be a simple table name or a full path name. A table name is preferred for portability to other SQL data sources.

col_name is an optional list of column names giving the name and order of the columns whose values are specified in the Values clause. If you omit *col_name*, the value expressions (*expr*) must provide values for all columns defined in the file and must be in the same order that the columns are defined for the file.

expr is the list of expressions giving the values for the columns of the new record. Usually, the expressions are constant values for the columns. Character string values must be enclosed in single (') or double (") quotation marks, date values must be enclosed in braces {}, and logical values that are letters must be enclosed in periods (for example, .T. or .F.).

An example of an Insert statement that uses a list of expressions is:

```
INSERT INTO emp (last_name, first_name, emp_id, salary, hire_date)
VALUES ('Smith', 'John', 'E22345', 27500, {4/6/1999})
```

Each Insert statement adds one record to the database table. In this case a record has been added to the employee database table, emp. Values are specified for five columns. The remaining columns in the table are assigned a blank value, meaning NULL.

select_statement is a query that returns values for each *col_name* value specified in the column name list. Using a Select statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single Insert statement.

An example of an Insert statement that uses a Select statement is:

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept, salary)
SELECT first_name, last_name, emp_id, dept, salary from emp
WHERE dept = 'D050'
```

In this type of Insert statement, the number of columns to be inserted must match the number of columns in the Select statement. The list of columns to be inserted must correspond to the columns in the Select statement just as it would to a list of value expressions in the other type of Insert statement. That is, the first column inserted corresponds to the first column selected; the second inserted to the second, and so forth.

The size and data type of these corresponding columns must be compatible. Each column in the Select list should have a data type that the driver accepts on a regular Insert/Update of the corresponding column in the Insert list. Values are truncated when the size of the value in the Select list column is greater than the size of the corresponding Insert list column.

The Select statement is evaluated before any values are inserted. This query cannot be made on the table into which values are inserted.

Update Statement

The Update statement is used to change rows in a database file. The form of the Update statement supported for flat-file drivers is:

```
UPDATE table_name SET col_name = expr, ...
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

table_name can be a simple table name or a full path name. A table name is preferred for portability to other SQL data sources.

col_name is the name of a column whose value is to be changed. Several columns can be changed in one statement.

expr is the new value for the column. The expression can be a constant value or a subquery. Character string values must be enclosed with single (') or double (") quotation marks, date values must be enclosed by braces {}, and logical values that are letters must be enclosed by periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The Where clause (any valid clause described in Select Statement on page 454) determines which rows are to be updated.

The Where Current Of *cursor_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor_name* is positioned to be updated. This is called a "positioned update." You must first execute a Select...For Update statement with a named cursor and fetch the row to be updated.

An example of an Update statement on the emp table is:

```
UPDATE emp SET salary=32000, exempt=1
WHERE emp_id = 'E10001'
```

The Update statement changes every record that meets the conditions in the Where clause. In this case, the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the emp table, only one record is updated.

An example using a subquery is:

```
UPDATE emp SET salary = (SELECT avg(salary) FROM emp)
WHERE emp_id = 'E10001'
```

In this case, the salary is changed to the average salary in the company for the employee having employee ID E10001.

Delete Statement

The Delete statement is used to delete rows from a database table. The form of the Delete statement supported for flat-file drivers is:

```
DELETE FROM table_name
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

table_name can be a simple table name or a full path name. A table name is preferred for portability to other SQL data sources.

The Where clause determines which rows are to be deleted. If you include only the keyword Where, all rows in the table are deleted, but the file is left intact.

The Where Current Of *cursor_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor_name* is positioned to be deleted. This is called a "positioned delete." You must first execute a Select...For Update statement with a named cursor and fetch the row to be deleted.

An example of a Delete statement on the emp table is:

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

Each Delete statement removes every record that meets the conditions in the Where clause. In this case, every record having the employee ID E10001 is deleted. Because employee IDs are unique in the employee table, at most, one record is deleted.

Reserved Keywords

The following words are reserved for use in SQL statements. If they are used for file or column names in a database that you use, you must enclose them in double (") quotation marks in any SQL statement where they appear as file or column names.

ALL	FROM	LIKE	OR
AND	FULL	NATURAL	ORDER
BETWEEN	GROUP	NOT	RIGHT
COMPUTE	HAVING	NULL	UNION
CROSS	INNER	ON	WHERE
DISTINCT	INTO	OPTIONS	
FOR	LEFT	OR	

SQL Functionality for the Impala Wire Protocol Driver

The DataDirect Connect XE *for* ODBC and DataDirect Connect64 XE *for* ODBC for Impala Wire Protocol driver support an extended set of SQL 92, in addition to the syntax for Impala SQL, which is a subset of SQL 92.

Refer to the Impala Language Reference documentation for information about using Impala SQL.

Data Definition Language (DDL)

The Impala Wire Protocol driver supports a broad set of DDL, including (but not limited to) the following:

- CREATE Database and DROP Database
- CREATE Table and DROP Table
- ALTER Table and Alter Partition statements

Refer to the Impala Language Reference documentation for information about using Impala SQL.

Selecting Data With the Driver

Select List

The following sections apply to the way the Select list can be used with the driver.

Column Name Qualification

A column can only be qualified with a single name. Furthermore, a table can be qualified with a database (ODBC schema) name in the FROM clause, and in some cases, must also be aliased. Aliasing may not be necessary if the database qualifier is not the current database.

The driver can work around these limitations using the Remove Column Qualifiers connection option.

- If set to *1*, the driver removes three-part column qualifiers and replaces them with alias.column qualifiers.
- If set to *0*, the driver does not do anything with the request.

Suppose you have the following ANSI SQL query:

```
SELECT schema.table1.col1,schema.table2.col2 FROM schema.table1,schema.table2
WHERE schema.table1.col3=schema.table2.col3
```

If the Remove Column Qualifiers connection option is enabled, the driver replaces the three-part column qualifiers:

```
SELECT  table1.col1,
table2.col2 FROM schema.table1 table1 JOIN  schema.table2 table2
WHERE  table1.col3 =  table2.col3
```

From Clause

LEFT, RIGHT, and FULL OUTER JOINs are supported, as are LEFT SEMI JOINs and CROSS JOINs using the equal comparison operator, as shown in the following examples:

```
SELECT a.* FROM a JOIN b ON (a.id = b.id AND a.department = b.department)

SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON
(c.key = b.key2)

SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key)
  WHERE a.ds='2009-07-07' AND b.ds='2009-07-07'
```

However, the following syntax fails because of the use of non-equal comparison operators.

```
SELECT a.* FROM a JOIN b ON (a.id <> b.id)
```

Impala SQL does not support join syntax in the form of a comma-separated list of tables. The driver, however, overcomes this limitation by translating the SQL into Impala SQL, as shown in the following examples.

ANSI SQL 92 Query	Impala SQL Translation
`SELECT * FROM t1, t2 WHERE a = b`	`SELECT * FROM t1 t1 JOIN t2 t2 WHERE a = b`
`SELECT * FROM t1 y, t2 x WHERE a = b`	`SELECT * FROM t1 y JOIN t2 x WHERE a = b`

ANSI SQL 92 Query	Impala SQL Translation
`SELECT * FROM t2, (SELECT * FROM t1) x`	`SELECT * FROM t2 t2 JOIN (SELECT * FROM t1 t1) x`

Group By Clause

The Group By clause is supported, with the following Entry SQL level restrictions:

- The COLLATE clause is not supported.
- SELECT DISTINCT is not supported.

Having Clause

The Having Clause is supported, with the following Entry SQL level restriction: a GROUP BY clause is required.

Order By Clause

The Order By clause is supported, with the following Entry SQL level restrictions:

- An integer sort key is not allowed.
- The COLLATE clause is not supported.
- A LIMIT clause or a default ORDER BY limit for result sets is required.

Default ORDER BY Limit

Impala will not return result sets for statements containing an order by clause unless a limit clause is included or a default limit is specified for result sets. A default limit can be set in the server; however, this limits the result sets for all queries, not just statements containing the ORDER BY clause.

The driver can work around these limitations by using the Default Order By Limit connection option.

where:

x

is a positive integer that represents maximum number of rows returned.

If set to *x*, the number of rows returned by a SQL statement containing an ORDER BY clause are limited to the specified number of rows for the session. To override this value, specify a new value in a LIMIT clause in the statement that is being executed.

If set to *-1*(disabled), there is no default limit the number of rows returned by a statement containing an ORDER BY clause. The application must limit the number of rows returned by SQL statements that contain an ORDER BY clause, or Impala will return an error.

For Update Clause

Not supported in this release. If present, the driver strips the For Update clause from the query.

Set Operators

Supported, with the following restrictions: INTERSECT or EXCEPT are not supported.

Subqueries

A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level query is called a Select statement, and a query nested within a Select statement is called a subquery.

Subqueries are supported, with the following Entry SQL level restriction: subqueries can only exist in the FROM clause, that is, in a derived table. In the following example, the second Select statement is a subquery:

```
SELECT * FROM (SELECT * FROM t1 UNION ALL SELECT * FROM t2) sq
```

Although Impala currently does not support IN or EXISTS subqueries, you can efficiently implement the semantics by rewriting queries to use LEFT SEMI JOIN.

SQL Expressions

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. You can use expressions in the Where and Having clauses of Select statements.

Expressions enable you to use mathematical operations as well as character string manipulation operators to form complex queries.

Valid expression elements are:

- Constants on page 472
- Numeric Operators on page 472
- Character Operator on page 473
- Relational Operators on page 473
- Logical Operators on page 474
- Functions on page 474

Constants

Impala uses binary literals for internal functions. Although the driver supports binary literals, no useful information is returned.

Numeric Operators

You can use a numeric operator in an expression to negate, add, subtract, multiply, and divide numeric values. The result of this operation is also a numeric value. The + and - operators are also supported in date/time fields to allow date arithmetic.

The following table lists the supported arithmetic operators.

Table 54: Numeric Operators

Entry SQL Level Operator	Impala SQL Operator
N/A	% (Mod)
N/A	& (bitwise AND)
*	Supported
+	Supported
-	Supported
/	Supported
N/A	^ (XOR)

Character Operator

The concatenation operator (||) is not supported; however, the CONCAT function is supported by Impala SQL.

```
SELECT CONCAT('Name is', '(ename FROM emp)')
```

Relational Operators

Relational operators compare one expression to another.

The following table lists the supported relational operators.

Table 55: Relational Operators Supported with Impala

Entry SQL Level Operator	Support in Impala SQL
<>	Supported
<	Supported
<=	Supported
=	Supported
<=>	Supported
>	Supported
>=	Supported
IS [NOT] NULL	Supported
[NOT] BETWEEN x AND y	Supported
[NOT] IN	Supported

Entry SQL Level Operator	Support in Impala SQL
EXISTS	Supported
[NOT] LIKE	Supported, except that no collate clause is allowed
RLIKE	Supported
REGEXP	Supported

Logical Operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 56: Logical Operators

Operator	Support in Impala HQL
NOT !	Supported
AND &&	Supported
OR ||	Supported

Functions

The following tables show how SQL-92 functions are supported in the Impala Query Language. Additional methods may be supported with ODBC Escapes.

Table 57: Set Functions Supported

Set Function	Support in Impala SQL
Count	Supported
AVG	Supported
MIN	Supported
MAX	Supported
SUM	Supported
DISTINCT	Supported
ALL	Supported

Table 58: Numeric Functions Supported

Numeric Function	Support in Impala SQL
CHAR_LENGTH CHARACTER_LENGTH	Not supported. Use LENGTH(string) instead.
Position...In	Not supported
BIT_LENGTH(s)	Not supported
OCTET_LENGTH(str)	Not supported
EXTRACT TIMEZONE_HOUR FROM	Not supported
EXTRACT TIMEZONE_MINUTE FROM	Not supported

Table 59: String Functions Supported

String Function	Support in Impala SQL
Substring	Supported
Convert … using	Not supported
TRIM	Supported.
Leading	Not supported. Use LTRIM.
Trailing	Not supported. Use RTRIM.
Both	Not supported (default behavior of TRIM)

Table 60: Date/Time Functions Supported

Date/Time Function	Support in Impala SQL
CURRENT_DATE( )	Not supported[24]
CURRENT_TIME( )	Not supported[24]
CURRENT_TIMESTAMP	Not supported. Use UNIX_TIMESTAMP()[24].

Table 61: System Functions Supported

System Function	Support in Impala SQL
CASE . . . END	Supported.
COALESCE	Supported.

[24] Supported by ODBC Escapes

System Function	Support in Impala SQL
NULLIF	Not supported.[24]
CAST	Supported.

Restrictions

This section describes some of the functional restrictions of Impala.

Insert and Update Restrictions

Impala does not support row level INSERT, UPDATE, and DELETE. Impala-specific syntax provides limited INSERT support:

- INSERT INTO is supported.
- INSERT OVERWRITE is supported.
- The name of the target table for an Insert must be prefixed by `table`, for example, `INSERT INTO table gtable …`

Column lists are not supported. Values for all columns must be specified.

Stored Procedures

Impala has no concept of stored procedures. Therefore, they are not supported by the driver.

Views

Impala has no concept of views. Therefore, they are not supported by the driver.

Other Restrictions

The Impala server has the following restrictions:

- Column values and parameters are always nullable
- No ROWID support
- No support for synonyms
- Primary and foreign keys are not supported.
- The length of a SQL string is limited to 2 GB.
- Support for indexes is incomplete.