Progress DataDirect for ODBC for Oracle Wire Protocol Driver

User's Guide

Release 8.0.2

Copyright

Table of Contents

Welcome to the Progress DataDirect for ODBC Oracle Wire Protocol Driver....11

What's new in this release?....12
Driver requirements....16
Installing and setting up the driver (Windows)....18
Installing and setting up the driver (UNIX/Linux)....20
ODBC compliance....22
Version string information....22
getFileVersionString function....24
Data types....24
XMLType....25
Retrieving data type information....27
SQL support....28
Additional information....29
Contacting Technical Support....29

Tutorials....31

The Example application....31
Tableau (Windows only)....33
Microsoft Excel (Windows only)....33
Power BI (Windows only)....35

Using the driver....37

Configuring and connecting to data sources....38
Configuring the product on UNIX/Linux....38
Data source configuration through a GUI....48
Using a connection string....88
Password Encryption Tool (UNIX/Linux only)....89
Using a logon dialog box....90
Performance considerations....92
Using LDAP....95
Connecting through a proxy server....99
Oracle Connection Manager....100
Support for Oracle RAC....101
XA interface support....101
MTS support....104
OS authentication....104

Isolation and lock levels supported....104
Unicode support....105
Using parameter arrays....105
Support of materialized views....105
Stored procedure results....105
Unexpected characters....106
Using failover....107
Connection failover....108
Extended connection failover....109
Select connection failover....110
Guidelines for primary and alternate servers....111
Using client load balancing....111
Using connection retry....112
Configuring failover-related options....113
Configuring failover using the TNSNAMES.ORA file....116
Using client information....117
How databases store client information....118
Storing client information....118
Using security....119
Authentication....119
Data encryption across the network....124
Data encryption and integrity....124
Summary of security-related options....136
Using DataDirect Connection Pooling....141
Creating a connection pool....142
Adding connections to a pool....142
Removing connections from a pool....142
Handling dead connections in a pool....143
Connection pool statistics....144
Summary of pooling-related options....144
Using DataDirect Bulk Load....145
Bulk export and load methods....146
Exporting data from a database....147
Bulk loading to a database....148
The bulk load configuration file....149
Sample applications....152
Character set conversions....152
External overflow files....152
Limitations....153
Summary of related options for DataDirect Bulk Load....153
Using bulk load for batch inserts....154
Determining the bulk load protocol....155
Limitations....155
Summary of related options for bulk load for batch inserts....156
Persisting a result set as an XML data file....156

Using the Windows XML persistence demo tool....157
Using the UNIX/Linux XML persistence demo tool....158
Packet logging....159

Connection option descriptions....163

Accounting Info....172
Action....172
Alternate Servers....173
Application Name....174
Application Using Threads....174
Array Size....175
Authentication Method....176
Batch Size....177
Batch Failure Returns Error....178
Bind Params As Unicode....178
Bulk Binary Threshold....179
Bulk Character Threshold....180
Bulk Options....181
Cached Cursor Limit....182
Cached Description Limit....182
Catalog Functions Include Synonyms....183
Catalog Options....184
Client Host Name....184
Client ID....185
Client User....186
Connection Pooling....187
Connection Reset....187
Connection Retry Count....188
Connection Retry Delay....189
Credentials Wallet Entry....190
Credentials Wallet Path....191
Crypto Protocol Version....192
CryptoLibName....192
Data Integrity Level....194
Data Integrity Types....195
Data Source Name....196
Default Buffer Size for Long/LOB Columns (in Kb)....196
Describe at Prepare....197
Description....197
Edition Name....198
Enable Bulk Load....199
Enable FIPS....200
Enable N-CHAR Support....201
Enable Scrollable Cursors....202

Enable Server Result Cache....202
Enable SQLDescribeParam....203
Enable Static Cursors for Long Data....204
Enable Timestamp with Timezone....204
Encryption Level....205
Encryption Method....206
Entra Access Token....207
Encryption Types....208
Failover Granularity....209
Failover Mode....210
Failover Preconnect....210
Fetch TSWTZ as Timestamp....211
Field Delimiter....212
GSS Client Library....213
Host....213
Host Name In Certificate....214
IANAAppCodePage....215
Impersonate User....216
Initialization String....217
Key Password....217
Key Store....218
Key Store Password....219
LDAP Crypto Protocol Version....219
LDAP Distinguished Name....220
LDAP Encryption Method....221
LDAP Key Store....222
LDAP Trust Store....223
LDAP Validate Server Certificate....224
Load Balancing....224
LoadBalance Timeout....225
LOB Prefetch Size....226
Local Timezone Offset....227
Lock Timeout....227
Login Timeout....228
Max Pool Size....229
Min Pool Size....229
Module....230
OpenSSLConfigFile....231
OpenSSLProviderPath....232
Password....232
Port Number....233
Proxy Host....234
Proxy Mode....234
Proxy Password....235
Proxy Port....236

Proxy User....237
PRNGSeedFile....238
PRNGSeedSource....239
Procedure Returns Results....240
Program ID....241
Query Timeout....242
Record Delimiter....242
Report Codepage Conversion Errors....243
Report Recycle Bin....244
SDU Size....245
Server Name....246
Server Process Type....246
Service Name....247
SID....248
SSLLibName....249
Support Binary XML....250
TCP Keep Alive....251
Timestamp Escape Mapping....252
TNSNames File....252
Trust Store....254
Trust Store Password....255
Use Current Schema for SQLProcedures....255
User Name....256
Validate Server Certificate....257
Wallet Password....257
Wire Protocol Mode....258

1

Welcome to the Progress DataDirect for ODBC Oracle Wire Protocol Driver

The 32-and 64-bit Progress® DataDirect® for ODBC Oracle™ Wire Protocol drivers support ODBC connectivity to:

- Oracle Database
- Oracle Autonomous Data Warehouse Cloud
- Oracle Autonomous Transaction Processing Cloud
- Oracle Database Cloud Service
- Oracle Database Exadata Cloud Service

The drivers are supported in Windows, UNIX, and Linux environments.

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

For details, see the following topics:

- What's new in this release?
- Driver requirements
- Installing and setting up the driver (Windows)
- Installing and setting up the driver (UNIX/Linux)

- ODBC compliance
- Version string information
- Data types
- SQL support
- Additional information
- Contacting Technical Support

What's new in this release?

Support and Certifications

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

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

Changes since 8.0.2 GA

- **Driver Enhancements**
  - The default version of the OpenSSL library has been upgraded to 3.5.6. As part of this upgrade, earlier version of the OpenSSL 3.0 library continues to be supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files:
    - Windows: `ivopenssl.dll` and `ddopenssl.dll`
    - Unix: `ivopenssl.so` and `ddopenssl.so`
  - The driver has been enhanced to support both unencrypted and encrypted connections to the LDAP server. After connecting to the LDAP server, the driver retrieves the necessary connection details from it and then connects to the database server. See Using LDAP on page 95 for details.
  - The driver has been enhanced to support authentication using Microsoft Entra ID (Azure Active Directory) access tokens. You can configure this authentication using the updated Authentication Method (AuthenticationMethod) connection option and the new Entra Access Token (EntraAccessToken) connection option. See Entra ID access token authentication on page 123 for details.
  - The driver is now compiled with a Visual Studio 2022 compiler for the Windows platforms. As a result, you must have Microsoft Visual C/C++ runtime version 14.40.33810 or higher on your machine to run the driver.
  - The driver has been enhanced to support the TLSv1.3 cryptographic protocol. As part of this enhancement, the default cryptographic protocols enabled by the driver have been updated to TLSv1.3 and TLSv1.2. See Crypto Protocol Version on page 192 for more information.
  - The default version of the OpenSSL library has been upgraded to 3.0. As part of this upgrade, earlier versions of the OpenSSL library are no longer supported to provide the best protection for your data. The upgrade is available in the following OpenSSL library files: *xx*`openssl30.dll` (for Windows) and *xx*`openssl30.so` [`.sl`] (for UNIX/Linux).

The OpenSSL 3.0 library uses a set of shared libraries called providers to implement different types of cryptographic algorithms. The driver supports the following OpenSSL 3.0 providers: FIPS and default. See TLS/SSL server authentication on page 125 and TLS/SSL client authentication on page 130 for details.

- The driver has been enhanced to support the Windows certificate store for TLS/SSL server authentication. See TLS/SSL server authentication on page 125 for details.
- The driver has been enhanced to support TLS/SSL server authentication for the applications deployed in a serverless environment. The driver stores the TLS/SSL certificates in memory and lets applications use TLS/SSL server authentication without storing the truststore file on the disk. To use this enhancement, specify the content of the certificate in the refreshed Trust Store (`Truststore`) connection option or the new `SQL_COPT_INMEMORY_TRUSTSTORECERT` pre-connection attribute. See Trust Store on page 254 and Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 128 for details.
- The driver has been enhanced with the new BatchFailureReturnsError option, which determines the behavior of the driver when encountering an error in a parameter array insert with bulk load disabled. For details, see Batch Failure Returns Error on page 178.
- 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 89 for details.
- The driver has been enhanced to support distributed transactions. It implements the XA interface to enable support for distributed transactions. For details, see XA interface support on page 101.
- The driver has been enhanced to support extended connection failover for the connections established using the `TNSNAMES.ORA` file. See Configuring failover using the TNSNAMES.ORA file on page 116 for details.
- The driver has been enhanced to include timestamp in the internal packet logs by default. If you want to disable the timestamp logging in packet logs, set `PacketLoggingOptions=1`. The internal packet logging is not enabled by default. To enable it, set `EnablePacketLogging=1`.
- The driver has been enhanced to support Oracle Wallet Password Stores. When this feature is enabled, the driver retrieves database credentials from an Oracle Wallet to be used for authentication to the server. The driver has also been enhanced with the new Credentials Wallet Entry (CredentialsWalletEntry), Credentials Wallet Path (CredentialsWalletPath), Wallet Password (CredentialsWalletPassword) options, which allow you to configure this feature. See Oracle Wallet Password Store on page 122 for details.
- On Windows and UNIX/Linux, the driver has been enhanced to support using connection information stored in an LDAP entry to establish a connection. You can configure the driver to use LDAP with the new LDAP Distinguished Name (LDAPDistinguishedName) option and refreshed Host (HostName) and Port Number (PortNumber) options. For details, see Using LDAP on page 95.
- The Driver Manager for UNIX/Linux has been enhanced to support setting the Unicode encoding type for applications on a per connection basis. By passing a value for the SQL_ATTR_APP_UNICODE_TYPE attribute using SQLSetConnectAttr, your application can specify the encoding at connection. This allows your application to pass both UTF-8 and UTF-16 encoded strings with a single environment handle.

  Refer to the "Driver Manager and Unicode encoding on UNIX/Linux" in *Progress DataDirect for ODBC Drivers Reference* for details.
- On Windows and UNIX/Linux, the driver has been enhanced to support connecting through Oracle Connection Manager. See Oracle Connection Manager on page 100 for details.
- The driver has been enhanced to support the following new statement attributes that allow you to override connection option settings for an individual statement:
  - SQL_ATTR_BULK_LOAD_ENABLED statement attribute overrides the EnableBulkLoad option
  - SQL_ATTR_IANA_APP_CODE_PAGE statement attribute overrides the IANAAppCodePage option

See Enable Bulk Load on page 199 and IANAAppCodePage on page 215 for details.

- On Windows and UNIX/Linux, the driver has been enhanced to support connecting to a proxy server through an HTTP connection. HTTP proxy support is configurable with five new connection options. See Proxy Host on page 234, Proxy Mode on page 234, Proxy Password on page 235, Proxy Port on page 236, and Proxy User on page 237 for details.
- The driver has been enhanced with the new Impersonate User connection option that allows you to specify the proxy user ID used for impersonation. The user ID specified using this option determines your permissions and identity when executing queries. See Impersonate User on page 216 for details.
- The driver has been enhanced to support using the default Service Name or SID specified in the server-side `listener.ora` file. See Service Name on page 247, SID on page 248, and TNSNames File on page 252 for details.
- The driver has been enhanced to support Oracle Database Vault.
- The driver has been enhanced to support the Oracle Database Exadata Cloud Service.

- **Changed Behavior**
  - The TLSv1.1 and TLSv1.0 cryptographic protocols are now disabled by default and have been removed as selectable values for the Crypto Protocol Version (CryptoProtocolVersion) option on the Setup dialog. These protocols are no longer considered secure and, therefore, are no longer recommended for use. However, the driver still supports TLSv1.1 and TLSv1.0 for legacy servers that do not support more secure protocols. See Crypto Protocol Version on page 192 for more information.
  - The valid value for the Authentication Method (AuthenticationMethod) option for retrieving credential information from Oracle Wallet password stores has been changed from `14` to `16`. To support existing configurations of the driver, the original value, `14`, will continue to be supported for this version of the driver. See Authentication Method on page 176 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.
  - On the GUI, proxy-related options have been moved from the General tab to the new Proxy tab.
  - The setting of the Array Size option can now be overridden by specifying the number of rows to fetch using the SQL_ATTR_ROW_ARRAY_SIZE statement attribute. See Array Size on page 175 for details.

Changes for 8.0.2 GA

- **Driver Enhancements**
  - Support for Oracle Wallet, including:
    - Oracle Wallet SSL Authentication
    - Using Oracle Wallet as a keystore or truststore for SSL data encryption.

    See Oracle Wallet SSL Authentication on page 121 and Using Oracle Wallet as a keystore on page 135 for details.
  - The driver has been certified to use Oracle Internet Directory as a means to store authentication information. See Oracle Internet Directory (OID) on page 121 for details.
  - The Oracle driver has been enhanced to support the following new data integrity algorithms for Oracle 12c and higher: SHA256, SHA384, SHA512. To use these algorithms, specify their values using the Data Integrity Types connection option and enable data integrity checks with the Data Integrity Level connection option. See Data Integrity Types on page 195 and Data Integrity Level on page 194 for details.
  - The maximum supported length of identifiers has been increased to 128 bytes when connecting to Oracle 12c R2 (12.2) databases. This change has been implemented to reflect the new maximum length supported by the server.
- **Changed Behavior**
  - The default value for the Data Integrity Types connection option has changed to the following:

    `MD5,SHA1,SHA256,SHA384,SHA512`

    See Data Integrity Types on page 195 for details.

Changes for 8.0.1 GA

- **Driver Enhancements**
  - Support for the Oracle 12 and 12a authentication protocols, which provide improved security.
  - Support for returning implicit result sets from stored procedures.
  - The driver is now compiled using Visual Studio 2015 for improved security.
  - The new SDU Size connection option allows you to specify the size in bytes of the Session Data Unit (SDU) that the driver requests when connecting to the server. See SDU Size on page 245 for details.
  - The new Support Binary XML connection option enables the driver to support XMLType with binary storage on servers running Oracle 12*c* and higher. See Support Binary XML on page 250 for details.
  - The new LOB Prefetch Size connection option allows you to specify the size of prefetch data the driver returns for BLOBs and CLOBs for Oracle database versions 12.1.0.1 and higher. With LOB prefetch enabled, the driver can return LOB meta-data and the beginning of LOB data along with the LOB locator during a fetch operation. This can have significant performance impact, especially for small LOBs which can potentially be entirely prefetched, because the data is available without having to go through the LOB protocol. See LOB Prefetch Size on page 226 for details.
- **Changed Behavior**
  - The Enable N-CHAR Support connection option has been deprecated, and the driver behavior has been updated to always provide support for the N-types NCHAR, NVARCHAR2 and NCLOB. For compatibility purposes, the EnableNcharSupport attribute can still be manually specified for this release, but will be deprecated in subsequent versions of the product. See Enable N-CHAR Support on page 201 and for details.

- The Enable Timestamp with Timezone connection option has been deprecated, and the driver behavior has been updated to always expose timestamps with timezones to the application. For compatibility purposes, the EnableTimestampwithTimezone attribute can still be manually specified for this release, but it will be deprecated in subsequent versions of the product. See Enable Timestamp with Timezone on page 204 for details.
- The default value for the Data Integrity Level connection option has been updated to `1` (Accepted). By default, a data integrity check can now be made on data sent between the driver and the database server, if the server request or requires it. This change allows the driver to connect to servers requiring Oracle Advanced Security data integrity checks using the default configuration. See Data Integrity Level on page 194 for details.
- The default value for the Encryption Level connection option has been updated to `1` (Accepted). By default, encryption is now used on data sent between the driver and the database server if the database server requests or requires it. This change allows the driver to connect to servers requiring Oracle Advanced Security encryption using the default configuration. See Encryption Level on page 205 for details.

Driver requirements

Data source and platform requirements

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

Windows requirements for 32-bit drivers

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

Windows requirements for 64-bit drivers

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

Linux requirements for 32-bit drivers

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

Linux requirements for 64-bit drivers

- The following processors are supported:
  - Intel Itanium II (IPF)
  - Intel and AMD processors
- 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)

AIX requirements for 32-bit and 64-bit drivers

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

HP-UX requirements for 32-bit drivers

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

HP-UX requirements for 64-bit drivers

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

Oracle Solaris requirements for 32-bit drivers

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

Oracle Solaris requirements for 64-bit drivers

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

Installing and setting up the driver (Windows)

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

To begin accessing data with the driver:

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

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

   c) Follow the prompts to complete installation.

   Note:

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

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

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

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

     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: To configure a new system data source, click **Add** to display a list of installed drivers. Select your driver and click **Finish** to display the driver Setup dialog box.

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

4. The General tab of the Setup dialog box appears by default. Provide values for the following essential connection options; then, click **Apply**:

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.

- **Host**: Type either the name or the IP address of the server to which you want to connect.
- **Port Number**: Type the port number of your Oracle listener. Check with your database administrator for the correct number.
- **SID**: Type the Oracle System Identifier that refers to the instance of Oracle running on the server. This option and the Service Name option are mutually exclusive. If the Service Name option is specified, do not specify this option.
- **Service Name**: Type the Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name, for example: `sales.us.acme.com`. This option and the SID option are mutually exclusive. If the SID option is specified, do not specify this option.

Note: If no values are specified for the SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.

Note: For demonstration purposes, this procedure uses the default authentication method, user ID and password (1 - Encrypt Password), to authenticate to the server. However, you can use any of the authentication methods supported by the driver. To know more about them, see "Authentication".

---

5. Set the values for any additional connection options that you want to configure. To view more options, click on the tabs on the dialog. See the following resources for additional information on optional features and functionality:
   - Authentication on page 119 provides configuration examples for the authentication methods supported by the driver.
   - Connection option descriptions on page 163 provides a complete list of supported options by the driver.
   - Data source configuration through a GUI on page 48 guides you through using the GUI to configure the driver.
   - Performance considerations on page 92 describes connection options that affect performance, along with recommended settings.

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

---

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. Provide values for the following connection options in it; then, click **OK**.

- **User Name**: Type your Oracle user name.
- **Password**: Type your Oracle password.

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

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

This completes the deployment of the driver.

See also

Using a connection string on page 88
Authentication on page 119

Installing and setting up the driver (UNIX/Linux)

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

To begin accessing data with the driver:

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

      `PROGRESS_DATADIRECT_ODBC_nn_platform_xx_INSTALL.bin`

   c) Follow the prompts to complete installation.

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

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

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

- Sets the ODBCINI environment variable to point to the path from the root directory to the system information file where your data source resides. For details, see "ODBCINI."

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

- **odbc.ini file**: You can begin using the driver immediately by editing the `odbc.ini` file in the installation directory with a text editor. The following demonstrates a data source definition with the minimal options used to authenticate using user ID and password (Encrypt Password) authentication.

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


[Oracle Wire Protocol]
Driver=<install_dir>/lib/xxora28.yy
...
Description=My Oracle Wire Protocol Data Source
...
HostName=YourOracleServer
...
LogonID=JohnD
...
Password=secret
...
PortNumber=1521
...
ServiceName=sales.us.acme.com
...
```

See Configuring a data source in the system information file on page 41 for more information.

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

- **Connection string**: The driver also supports using connection strings for DSN (data source name), File DSN, or DSN-less connections. See Using a connection string, and DSN-less connections, for more information. For examples of connection strings using different authentication methods, see Authentication .

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

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

- Connection option descriptions provides a complete list of supported options by functionality.

- Performance Considerations describes connection options that affect performance, along with recommended settings.

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

This completes the deployment of the driver.

ODBC compliance

The driver is compliant with the Open Database Connectivity (ODBC) specification and compatible with ODBC 3.8 applications. 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
- SQLProcedures
- SQLProcedureColumns
- SQLSetPos
- SQLTablePrivileges

Refer to "ODBC API and scalar functions" in the *Progress DataDirect for ODBC Drivers Reference* for additional information.

Version string information

The driver has a version string of the format:

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

or

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

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

XX.YY.ZZZZ(U*BBBB*)

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

XX.YY.ZZZZ

where:

XX is the major version of the product.

YY is the minor version of the product.

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

AAAA is the build number of the driver's base component.

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

For example:

```
08.00.0001 (b0001, u0002)
      |__|  |___|  |___|
     Driver  Base   Utl
```

On Windows, you can check the version string using the properties window of the driver file. First, right-click the driver `.dll` file and select **Properties**. Then, on the Properties window, select the **Details** tab. The product version field lists the version string.

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 drivers and ddtestlib for 64-bit drivers, is launched using a command-line and 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 Wire Protocol driver on Linux:

```
ivtestlib ivora28.so
```

returns:

```
08.00.0001 (B0002, U0001)
```

For example, for the Driver Manager on Linux:

```
ivtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

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

```
ddtestlib libodbc.so
```

returns:

```
08.00.0001 (U0001)
```

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

```
ivtestlib libivicu28.so
08.00.0001
```

Note: Only the HP-UX version of the tool requires specifying the full path for the test loading tool. The full path does not need to be specified for other platforms.

getFileVersionString function

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

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

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

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

Data types

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

Table 1: Oracle Data Types

Oracle	ODBC
BFILE[1]	SQL_LONGVARBINARY
BINARY DOUBLE	SQL_REAL
BINARY FLOAT	SQL_DOUBLE
BLOB[2]	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB[2]	SQL_LONGVARCHAR
DATE	SQL_TYPE_TIMESTAMP
INTERVAL DAY TO SECOND	SQL_VARCHAR
INTERVAL YEAR TO MONTH	SQL_VARCHAR
LONG	SQL_LONGVARCHAR

[1] Read-Only
[2] Supported in basic file and SecureFiles storage.

Oracle	ODBC
LONG RAW	SQL_LONGVARBINARY
NCHAR	SQL_WVARCHAR
NCLOB	SQL_WLONGVARCHAR
NVARCHAR2	SQL_WVARCHAR
NUMBER	SQL_DOUBLE
NUMBER (p,s)	SQL_DECIMAL
RAW	SQL_VARBINARY
TIMESTAMP	SQL_TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE	SQL_TIMESTAMP
TIMESTAMP WITH TIMEZONE	SQL_VARCHAR
VARCHAR2	SQL_VARCHAR
XMLType[3]	SQL_LONGVARCHAR

The Oracle Wire Protocol driver does not support any object types (also known as abstract data types). When the driver encounters an object type during data retrieval, it returns an Unknown Data Type error (SQL State HY000).

See "Retrieving data type information" for more information about data types.

See also

Retrieving data type information on page 27

XMLType

The driver supports tables containing columns whose data type is specified as XMLType, except those with object relational storage.

In the default configuration, the driver supports the XMLType with CLOB storage; however, beginning with Oracle 11.2.0.2, Oracle changed the default storage type from CLOB to Binary. To support the XMLType with binary storage on Oracle 12c or later, enable the Support Binary XML connection option (`SupportBinaryXML=1`).

As a result of the new default storage type, columns created simply as “XMLType” are not supported by the driver for database versions later than 11.2.0.1, but earlier than 12c. An attempt to obtain the value of such a column through the driver results in an error being returned. To avoid this error, change the XML storage type to CLOB or use the TO_CLOB Oracle function to cast the column.

[3] XMLType columns with object relational storage are not supported.

When inserting or updating XMLType columns, the data to be inserted or updated must be in the form of an XMLType data type. The database provides functions to construct XMLType data. The xmlData argument to xmltype( ) may be specified as a string literal.

Examples

If the XMLType column is created with the CLOB storage type, then the driver returns it without use of the special getClobVal function, that is, you can use:

```
SELECT XML_col FROM table_name...
```

instead of

```
SELECT XML_col.getClobVal()...
```

The following example illustrates using the CLOB storage type:

```
CREATE TABLE po_xml_tab(
  poid NUMBER(10),
  poDoc XMLTYPE
  )
  XMLType COLUMN poDoc
    STORE AS CLOB (
      TABLESPACE lob_seg_ts[4]
      STORAGE (INITIAL 4096 NEXT 4096)
      CHUNK 4096 NOCACHE LOGGING
    )
```

The next example illustrates how to create a table, insert data, and retrieve data when not using the CLOB storage type:

```
CREATE TABLE PURCHASEORDER (PODOCUMENT sys.XMLTYPE);
```

The PURCHASEORDER table contains one column—PODOCUMENT—with a data type of XMLType (`sys.XMLTYPE`). The next step is to insert one purchase order, created by the static function `sys.XMLTYPE.createXML`:

```
INSERT INTO PURCHASEORDER (PODOCUMENT) values (
sys.XMLTYPE.createXML(
'<PurchaseOrder>
   <Reference>BLAKE-2001062514034298PDT</Reference>
   <Actions>
      <Action>
         <User>KING</User>
         <Date/>
      </Action>
   </Actions>
   <Reject/>
   <Requester>David E. Blake</Requester>
   <User>BLAKE</User>
   <CostCenter>S30</CostCenter>
   <ShippingInstructions>
      <name>David E. Blake</name>
      <address>400 Oracle Parkway Redwood Shores, CA, 94065 USA</address>
      <telephone>650 999 9999</telephone>
    </ShippingInstructions>
   <SpecialInstructions>Air Mail</SpecialInstructions>
   <LineItems>
      <LineItem ItemNumber="1">
         <Description>The Birth of a Nation</Description>
         <Part Id="EE888" UnitPrice="65.39" Quantity="31"/>
      </LineItem>
```

[4] Note that the table space must be created before executing a statement similar to the one used in the example.

```
   </LineItems>
</PurchaseOrder>
'));
```

Use the getClobVal function to retrieve the data:

```
SELECT p.podocument.getClobVal() FROM PURCHASEORDER p;
```

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 all platforms, 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 int     columnSize;

    SQLLEN           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);
    }
}
```

See also

Data types on page 24

SQL support

The driver supports the core SQL grammar.

Additional information

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

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

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.

July 2020, Release 8.0.2 of the Progress DataDirect Connect for ODBC for Oracle Wire Protocol Driver, Version 0001

2

Tutorials

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

For details, see the following topics:

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

The Example application

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

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

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

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

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

Note:

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

To use the Example application:

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

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

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

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

Tableau (Windows only)

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

To use the driver to access data with Tableau:

1. Open Tableau. If the **Connect** menu does not open by default, select **Data > New Data Source** or the Add New Data Source button  to open the menu.
2. From the **Connect** menu, select **Other Databases (ODBC)**.
3. The **Other Databases (ODBC)** dialog appears. In the DSN field, select the data source you want to use from the drop-down menu. For example, **My DSN**. Then, click **Connect**. The Logon dialog appears pre-populated with the connection information you provided in your data source.
4. If required, type your user name and password; then, click **OK**. The Logon dialog closes. Then, click **Sign in** on the Other Databases (ODBC) dialog.
5. The **Data Source** window appears. By default, Tableau connects live, or directly, to your data. We recommend that you use the default settings to avoid extracting all of your data. However, if you prefer, you can import your data by selecting the **Extract** option at the top of the dialog.
6. Select the database and schema you want to use from their respective drop-down lists. The tables stored in the schema you selected are now available for selection in the Table field. To access data, you can either drag the tables into the canvas on the right or double-click the **New Custom SQL** button at the bottom of the window and then run SQL statements in the **Edit Custom SQL** window.

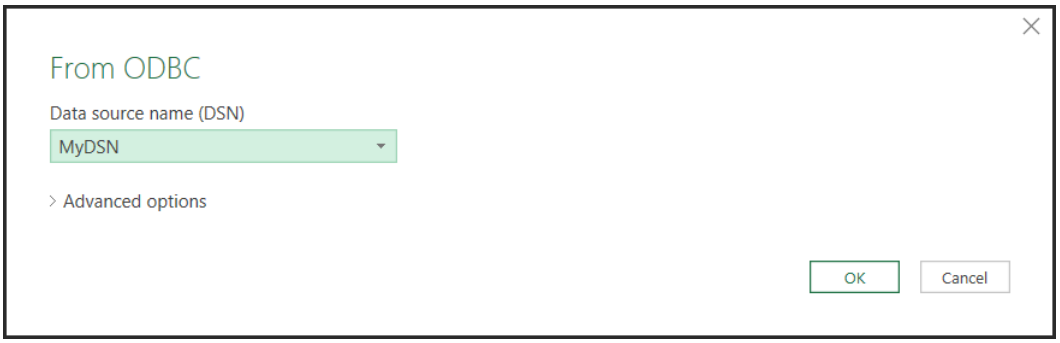
You have successfully accessed your data and are now ready to create reports with Tableau. For more information, refer to the Tableau product documentation at: http://www.tableau.com/support/help.

Microsoft Excel (Windows only)

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

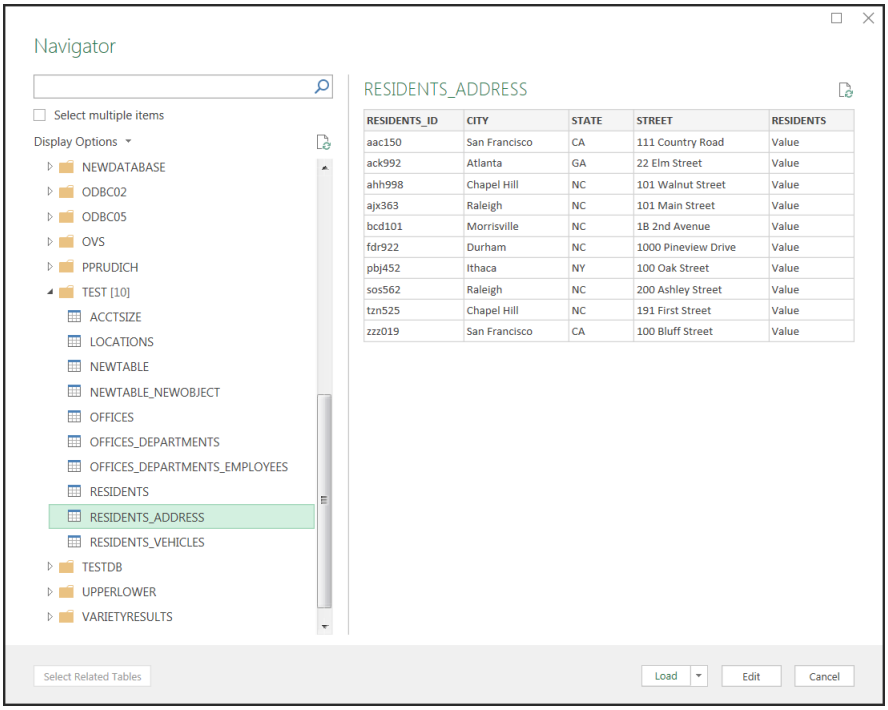
To use the driver to access data with Excel from the Data Connection Wizard:

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



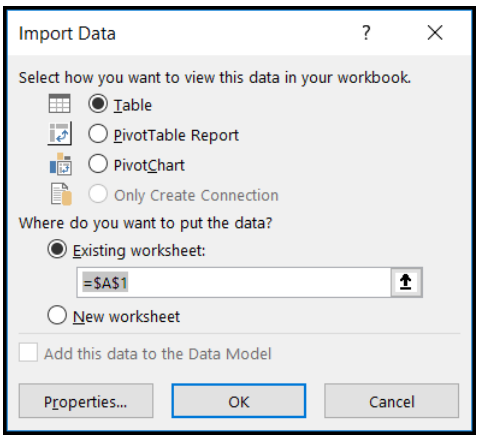
Select your data source from the Data Source Name (DSN) drop down; then, click **OK**.

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



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

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



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

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

Power BI (Windows only)

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

To use the driver to access data with Power BI:

1. Open the Power BI desktop application.
2. From the **Get data** menu, select **More**.
3. The **Get Data** window appears. Select **Other > ODBC**; then, click **Connect**.
4. The **From ODBC** dialog appears. Select your data source from the **Data source name (DSN)** drop-down list; then, click **OK**. To learn how to create a data source, see "Installing and setting up the driver (Windows)".
5. Enter authentication information when prompted. Once connected, the **Navigator** window displays schema and table information.
6. Select and load tables. Then, prepare your Power BI dashboard as desired.

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

See also

Installing and setting up the driver (Windows) on page 18

3

Using the driver

This chapter guides you through the configuring and connecting to data sources. In addition, it explains how to use the functionality supported by your driver.

For details, see the following topics:

- Configuring and connecting to data sources
- Performance considerations
- Using LDAP
- Connecting through a proxy server
- Support for Oracle RAC
- XA interface support
- MTS support
- OS authentication
- Isolation and lock levels supported
- Unicode support
- Using parameter arrays
- Support of materialized views
- Stored procedure results
- Unexpected characters

- Using failover
- Using client information
- Using security
- Using DataDirect Connection Pooling
- Using DataDirect Bulk Load
- Using bulk load for batch inserts
- Persisting a result set as an XML data file
- Packet logging

Configuring and connecting to data sources

After you install the driver, you configure data sources to connect to the database. Information that the driver needs to connect to a database is stored in a *data source*. The ODBC specification describes three types of data sources: user data sources, system data sources (not a valid type on UNIX/Linux), and file data sources. On Windows, user and system data sources are stored in the registry of the local computer. The difference is that only a specific user can access user data sources, whereas any user of the machine can access system data sources. On all platforms, file data sources, which are simply text files, can be stored locally or on a network computer, and are accessible to other machines. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

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

See also

Using a connection string on page 88
Connection option descriptions on page 163

Configuring the product on UNIX/Linux

UNIX®

This chapter contains specific information about using your driver in the UNIX and Linux environments.

See "Environment variables" for additional platform information.

See also

Environment variables on page 39

Environment variables

The first step in setting up and configuring the driver 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 Progress DataDirect *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.

ODBCINI

Setup installs in the product installation directory a default system information file, named odbc.ini, that contains data sources. See "Data source configuration on UNIX/Linux" for an explanation of the odbc.ini file. The system administrator can choose to rename the file and/or move it to another location. In either case, the environment variable ODBCINI must be set to point to the fully qualified path name of the odbc.ini file.

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

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

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

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

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

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

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

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

See also

Data source configuration on UNIX/Linux on page 41

ODBCINST

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

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

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

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

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

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

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

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

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

See also

DSN-less connections on page 45

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 "Configuring a data source in the system information file" for a description of the InstallDir keyword)

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

The next step is to test load the driver.

See also

Configuring a data source in the system information file on page 41

Data source configuration on UNIX/Linux

In the UNIX and Linux environments, a system information file is used to store data source information. Setup installs a default version of this file, called `odbc.ini`, in the product installation directory. This is a plain text file that contains data source definitions.

Configuring a data source in the system information file

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:

```
Oracle Wire Protocol=DataDirect Oracle 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, `[Oracle 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. See "Connection option descriptions" for descriptions of these attributes. See "Sample odbcinst.ini file" for sample data sources.

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

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

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

Note: This section is not included in the default `odbc.ini` file that is installed by the product installer. If you are using file data sources, you must add this section manually.

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

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

The IANAAppCodePage keyword defines the default value that the UNIX/Linux driver uses if individual data sources have not specified a different value. See "IANAAppCodePage" in "Connection option descriptions". The default value is `4`.

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

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

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

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

Note: If you are using only DSN-less connections through an `odbcinst.ini` file and do not have an `odbc.ini` file, then you must provide `[ODBC]` section information in the `[ODBC]` section of the `odbcinst.ini` file. The 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" 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*.

See also

Connection option descriptions on page 163
Sample odbcinst.ini file on page 46
File data sources on page 46
IANAAppCodePage on page 215
DSN-less connections on page 45

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 a 32-bit driver with the driver file name beginning with `iv`. A 64-bit driver file would be identical except that driver name would begin with `dd` and the list of data sources would include only the 64-bit drivers.

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

[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
AccountingInfo=
Action=
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=60000
AuthenticationMethod=1
BatchFailureReturnsError=0
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadFieldDelimiter=,
BulkLoadOptions=0
BulkLoadRecordDelimiter=
```

```
CachedCursorLimit=32
CachedDescLimit=0
CatalogIncludesSynonyms=1
CatalogOptions=0
ClientHostName=
ClientID=
ClientUser=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CredentialsWalletEntry=
CredentialsWalletPassword=
CredentialsWalletPath=
CryptoLibName=
CryptoProtocolVersion=TLSv1.3,TLSv1.2
DataIntegrityLevel=1
DataIntegrityTypes=MD5,SHA1,SHA256,SHA384,SHA512
DefaultLongDataBuffLen=1024
DescribeAtPrepare=0
EditionName=
EnableBulkLoad=0
EnableDescribeParam=0
EnableFIPS=0
EnableScrollableCursors=1
EnableServerResultCache=0
EnableStaticCursorsForLongData=0
EncryptionLevel=1
EncryptionMethod=0
EncryptionTypes=AES128,AES192,AES256,DES,3DES112,3DES168,RC4_40,RC4_56,RC4_128,RC4_256
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
GSSClient=native
HostName=<Oracle_server>
HostNameInCertificate=
ImpersonateUser=
InitializationString=
KeepAlive=0
KeyPassword=
KeyStore=
KeyStorePassword=
LDAPDistinguishedName=
LoadBalanceTimeout=0
LoadBalancing=0
LOBPrefetchSize=4000
LocalTimeZoneOffset=
LockTimeOut=-1
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
Module=

Password=
Pooling=0
PortNumber=<Oracle_server_port>
ProcedureRetResults=0
ProgramID=
PRNGSeedFile=/dev/random
PRNGSeedSource=0
ProxyHost=
ProxyMode=0
ProxyPassword=
ProxyPort=0
ProxyUser=
QueryTimeout=0
ReportCodePageConversionErrors=0
ReportRecycleBin=0
```

```
SDUSize=16384
ServerName=<server_name_in_tnsnames.ora>
ServerType=0
ServiceName=
SID=<Oracle_System_Identifier>
SupportBinaryXML=0
SSLLibName=
TimestampEscapeMapping=0
TNSNamesFile=<tnsnames.ora_filename>
TrustStore=
TrustStorePassword=
UseCurrentSchema=1
ValidateServerCertificate=1
WireProtocolMode=2


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

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

- **To modify a data source:**

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

  b) Modify the default 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 "Oracle Wire Protocol Attribute Names " table in the "Connection Options Descriptions" for other specific attribute values.

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

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

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

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

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

  c) Change the data source name in the copied data source definition to a new name. The data source name is between square brackets at the beginning of the definition, for example, `[Oracle 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 "Oracle Wire Protocol Attribute Names " table in the "Connection Option Descriptions" for other specific attribute values.

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

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

Important: The "Oracle Wire Protocol Attribute Names " table in the "Connection Option Descriptions" section lists both the long and short name of the attribute. When entering attribute names into odbc.ini, you must use the long name of the attribute. The short name is not valid in the odbc.ini file.

See also

Connection option descriptions on page 163

The example application

Progress DataDirect ships an application, named *example*, that is installed in the /samples/example subdirectory of the product installation directory. Once you have configured your environment and data source, use the example application to test passing SQL statements. To run the application, enter example and follow the prompts to enter your data source name, user name, and password.

If successful, a SQL> prompt appears and you can type in SQL statements, such as SELECT * FROM *table_name*. If *example* is unable to connect to the database, an appropriate error message appears.

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

Refer to "The example application" in *Progress DataDirect for ODBC Drivers Reference* for more information.

DSN-less connections

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

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

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

```
DataDirect 8.0 Oracle 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 "Connection option descriptions" for a description of the other keywords this section.

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

See also

ODBCINST on page 40
Connection option descriptions on page 163

Sample odbcinst.ini file

The following is a sample `odbcinst.ini`. All occurrences of ODBCHOME are replaced with your installation directory path during installation of the file. Commented lines are denoted by the # symbol. This sample shows a 32-bit driver with the driver file name beginning with `iv`; a 64-bit driver file would be identical except that driver names would begin with `dd`.

```
[ODBC Drivers]
DataDirect 8.0 Oracle Wire Protocol=Installed

[DataDirect 8.0 Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help/OracleHelp
Setup=ODBCHOME/lib/ivora28.so
SQLLevel=1

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

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

File data sources

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

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

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

```
[ODBC]
Driver=DataDirect 8.0 Oracle Wire Protocol
Port=1522
HostName=OraServer5
LogonID=JOHN
Servicename=SALES.US.ACME.COM
CatalogOptions=1
```

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

The file data source is accessed by specifying the FILEDSN= instead of the DSN= keyword in a connection string, as outlined in the ODBC specification. The complete path to the file data source can be specified in the syntax that is normal for the machine on which the file is located. For example, on Windows:

```
FILEDSN=C:\Program Files\Common Files\ODBC\DataSources\Oracleacct.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/Oracleacct.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 "Configuring a data source in the system information file" for details). If the [ODBC File DSN] setting is not defined, the Driver Manager uses the InstallDir setting in the [ODBC] section of the odbc.ini file. The Driver Manager does not support the SQLReadFileDSN and SQLWriteFileDSN functions.

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

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

See also

Configuring and connecting to data sources on page 38
DSN-less connections on page 45
Configuring a data source in the system information file on page 41

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);
    ```

Data source configuration through a GUI

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

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 Oracle data source:

1. 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.

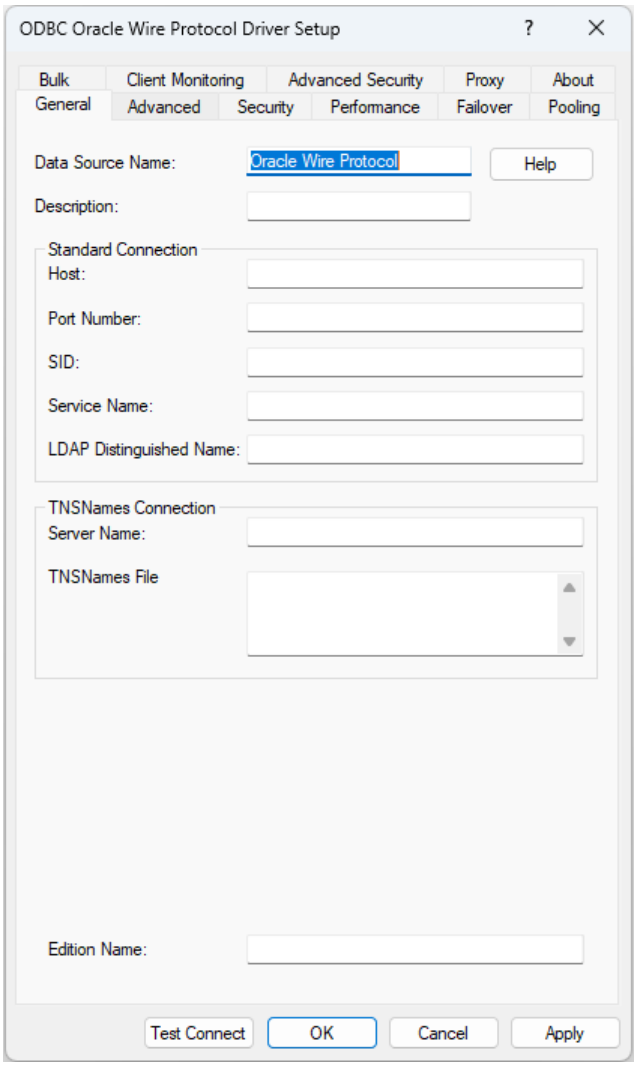
     If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.
   - **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** 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.

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

Figure 1: General tab



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

Connection Options: General	Description
Data Source Name on page 196	Specifies the name of a data source in your Windows Registry or `odbc.ini` file. **Default:** None
Description on page 197	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. **Default:** None

Connection Options: General	Description
Host on page 213	The name or the IP address of the server to which you want to connect. **Default:** None
Port Number on page 233	Specifies the port number of the server listener. **Default:** None
SID on page 248	The Oracle System Identifier that refers to the instance of Oracle running on the server. **Default:** None **Note:** This option is mutually exclusive with the LDAP Distinguished Name, Service Name, Server Name, and TNSNames File options. **Note:** If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.
Service Name on page 247	The Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name, for example: `sales.us.acme.com` **Default:** None **Note:** This option is mutually exclusive with the LDAP Distinguished Name, Service Name, Server Name, and TNSNames File options. **Note:** If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.

Connection Options: General	Description
LDAP Distinguished Name on page 220	Specifies the distinguished name for the LDAP entry that contains your connection information. Using an LDAP entry provides simplified maintenance by allowing you to centrally store and access connection information. LDAP entries specify the Host, Port Number, and Service Name or SID for the target database using the orclNetDescString attribute. **Note:** This option is mutually exclusive with the Host, Port Number, SID, and Service Name options. **Note:** If a value is specified for this option, the Host and Port Number options are used to specify the host and port number for the LDAP directory server.
Service Name on page 247	Specifies a net service name that exists in the `tnsnames.ora` file. The corresponding net service name entry in the `tnsnames.ora` file is used to obtain Host, Port Number, and Service Name or SID information. **Default:** None **Note:** This option is mutually exclusive with the LDAP Distinguished Name, Host, Port Number, SID, and Service Name options.
TNSNames File on page 252	Specifies the name of the `tnsnames.ora` file. **Default:** None **Note:** If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.
Edition Name on page 198	The name of the Oracle edition the driver uses when establishing a connection. Oracle 11*g* R2 and higher allows your database administrator to create multiple editions of schema objects so that your application can still use those objects while the database is being upgraded. This option is only valid for Oracle 11*g* R2 and higher databases and tells the driver which edition of the schema objects to use. **Default:** None

4. At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a logon dialog box" for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

5. To further configure your driver, click on the following tabs. The corresponding sections provide details on the fields specific to each configuration tab:
   - Advanced tab allows you to configure advanced behavior.
   - Security tab allows you to specify security data source settings.
   - Performance tab allows you to specify performance data source settings.
   - Failover tab allows you to specify failover data source settings.
   - Pooling tab allows you to specify connection pooling settings.
   - Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
   - Client Monitoring tab allows you to specify additional data source settings.
   - Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
   - Proxy tab allows you to specify settings for connecting through an HTTP proxy.
6. Click **OK**. When you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

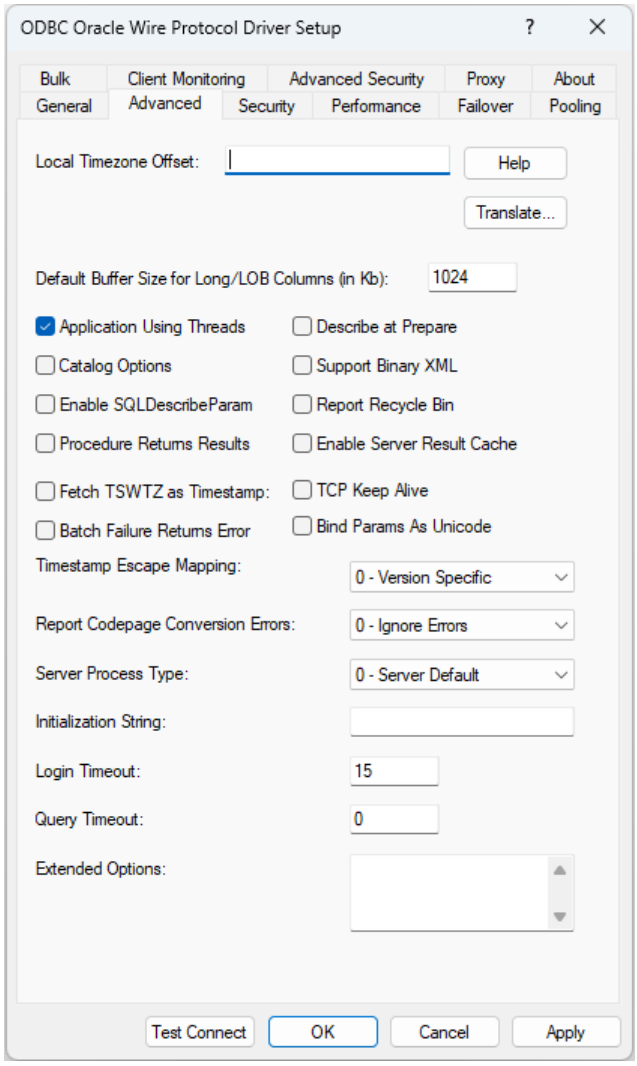
See also

Using a logon dialog box on page 90

Advanced tab

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

Figure 2: Advanced tab



Connection Options: Advanced	Description
Local Timezone Offset on page 227	A value to alter local time zone information. The default is an empty string, which means that the driver determines local time zone information from the operating system. If it is not available from the operating system, the driver defaults to using the setting on the Oracle server. Valid values are specified as offsets from GMT as follows: *(-)HH:MM*. For example, `-08:00` equals GMT minus 8 hours. **Default:** None

Connection Options: Advanced	Description
Default Buffer Size for Long/LOB Columns (in Kb) on page 196	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. **Default:** `1024`
Application Using Threads on page 174	Determines whether the driver works with applications using multiple ODBC threads. If enabled, the driver works with single-threaded and multi-threaded applications. If 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:** Enabled
Describe at Prepare on page 197	Determines whether the driver describes the SQL statement at prepare time. If enabled, the driver describes the SQL statement at prepare time. If disabled, the driver does not describe the SQL statement at prepare time. **Default:** Disabled
Catalog Options on page 184	Determines whether SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF. If enabled, the result column REMARKS (for the catalog functions SQLTables and SQLColumns) and the result column COLUMN_DEF (for the catalog function SQLColumns) return actual values. Enabling this option reduces the performance of your catalog (SQLColumns and SQLTables) queries. If disabled, SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF. **Default:** Disabled
Support Binary XML on page 250	Enables the driver to support XMLType with binary storage on servers running Oracle 12*c* and higher. If enabled, the driver supports XMLType with binary storage by negotiating server and client capabilities during connection time. As a result of this negotiation, decoded data associated with XMLType columns is returned in an in-line fashion without locators. If disabled, the driver does not support XMLType with binary storage and returns the error "This column type is not currently supported by this driver." **Default:** Disabled

Connection Options: Advanced	Description
Enable SQLDescribeParam on page 203	Determines whether the driver supports the SQLDescribeParam function, which allows an application to describe parameters in SQL statements and in stored procedure calls. If set to enabled, the driver supports SQLDescribeParam. If using Microsoft Remote Data Objects (RDO) to access data, you must use this value. If disabled, the driver does not support SQLDescribeParam and returns the error: `unimplemented function`. **Default:** Disabled
Report Recycle Bin on page 244	Determines whether support is provided for reporting objects that are in the Oracle Recycle Bin. If enabled, support is provided for reporting objects that are in the Oracle Recycle Bin. If disabled, the driver does not return tables contained in the recycle bin in the result sets returned from SQLTables and SQLColumns. Functionally, this means that the driver filters out any results whose Table name begins with BIN$. **Default:** Disabled
Procedure Returns Results on page 240	Determines whether the driver returns result sets from stored procedures/functions. If enabled, the driver returns result sets from stored procedures/functions. When set to 1 and you execute a stored procedure that does not return result sets, you will incur a small performance penalty. If disabled, the driver does not return result sets from stored procedures. **Default:** Disabled
Enable Server Result Cache on page 202	Determines whether the driver sets the RESULT_CACHE_MODE session parameter to FORCE. If enabled, the driver sets the RESULT_CACHE_MODE session parameter to FORCE. If disabled, the driver does not sets the RESULT_CACHE_MODE session parameter. **Default:** Disabled

Connection Options: Advanced	Description
Fetch TSWTZ as Timestamp on page 211	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. If 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 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:** Disabled
TCP Keep Alive on page 251	Specifies whether the driver enables TCPKeepAlive. TCPKeepAlive maintains idle TCP connections by periodically passing packets between the client and server. If disabled, the driver does not enable TCPKeepAlive. If enabled, the driver enables TCPKeepAlive. **Default:** Disabled
Batch Failure Returns Error on page 178	Determines the behavior of the driver when encountering an error in a parameter array insert with bulk load disabled (`EnableBulkLoad=0`). Note that this option applies only when operating in autocommit mode. If enabled, the driver returns `SQL_ERROR` and rolls back the operation when encountering an error in any of the parameter sets. If disabled, the driver returns `SQL_SUCCESS_WITH_INFO` and commits the rows that were successfully inserted prior to encountering the error. **Default:** Disabled
Timestamp Escape Mapping on page 252	Determines how the driver maps Date, Time, and Timestamp literals. If set to **0 - Oracle Version Specific**, the driver determines whether to use the TO_DATE or TO_TIMESTAMP function based on the version of the Oracle server to which it is connected. If the driver is connected to an 8.x server, it maps the Date, Time, and Timestamp literals to the TO_DATE function. If the driver is connected to a 9.x or higher server, it maps these escapes to the TO_TIMESTAMP function. If set to **1 - Oracle 8x Compatible**, the driver always uses the Oracle 8.x TO_DATE function as if connected to an Oracle 8.x server. **Default: 0 - Oracle Version Specific**

Connection Options: Advanced	Description
Report Codepage Conversion Errors on page 243	Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another. 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**
Server Process Type on page 246	Determines whether the connection is established using a shared or dedicated server process (dedicated thread on Windows). If set to **0 - Server Default**, the driver uses the default server process set on the server. If set to **1 - Shared**, the server process used is retrieved from a pool. The socket connection between the application and server is made to a dispatcher process on the server. This setting allows there to be fewer processes than the number of connections, reducing the need for server resources. Use this value when a server must handle a large number of connections. If set to **2 - Dedicated**, a server process is created to service only that connection. When that connection ends, so does the process (UNIX and Linux) or thread (Windows). The socket connection is made directly between the application and the dedicated server process or thread. When connecting to UNIX and Linux servers, a dedicated server process can provide significant performance improvement, but uses more resources on the server. When connecting to Windows servers, the server resource penalty is insignificant. Use this value if you have a batch environment with a low number of connections. **Default: 0 - Server Default**
Initialization String on page 217	A SQL command that is issued immediately after connecting to the database to manage session settings. **Default:** None
Login Timeout on page 228	The number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error. **Default:** 15
Query Timeout on page 242	The number of seconds for the default query timeout for all statements that are created by a connection. **Default:** 0 (the query does not time out.)

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.

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.

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Security tab on page 58 allows you to specify security data source settings.
- Performance tab on page 64 allows you to specify performance data source settings.
- Failover tab on page 68 allows you to specify failover data source settings.
- Pooling tab on page 71 allows you to specify connection pooling settings.
- Bulk tab on page 73 allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab on page 80 allows you to specify additional data source settings.
- Advanced Security tab on page 82 allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

See also

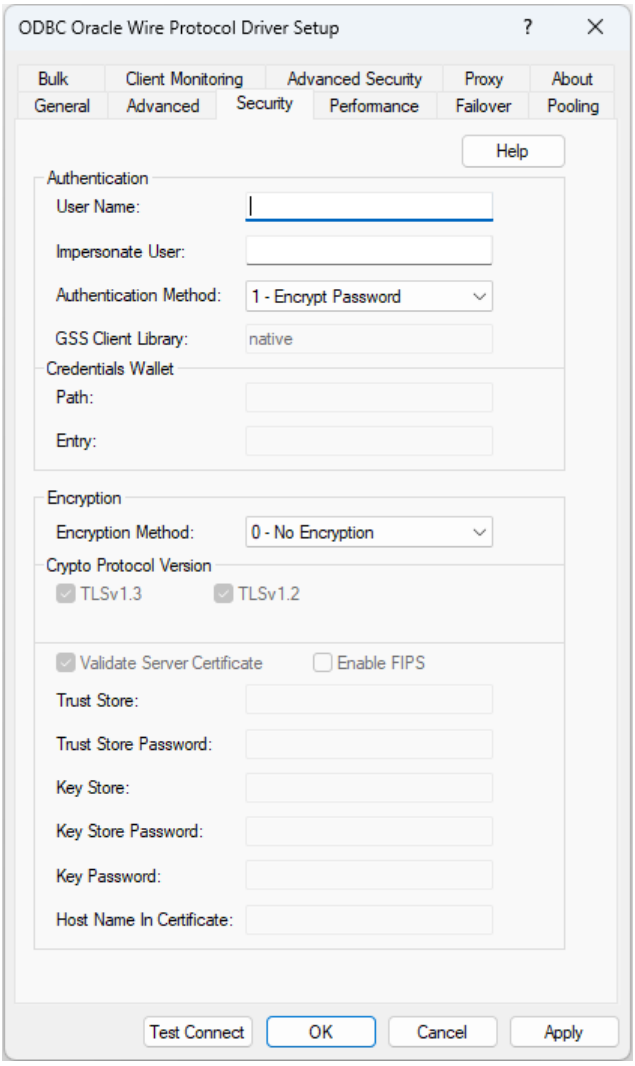
Data source configuration through a GUI on page 48

Security tab

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

See "Using security" for a general description of authentication and encryption and their configuration requirements.

Figure 3: Security tab



Connection Options: Security	Description
User Name on page 256	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. **Default:** None

Connection Options: Security	Description
Impersonate User on page 216	Specifies the proxy user ID used for impersonation. The value for Impersonate User determines your identity and permissions when executing queries. When a value is specified for this option, the driver authenticates according to the setting of the Authentication Method option; then, after establishing a connection, the driver attempts to reauthenticate as the destination user. Note that the administrator must grant CONNECT THROUGH permission to the authenticated user in order to impersonate the destination user; otherwise, an error is returned. **Default:** None
Authentication Method on page 176	Specifies the method the driver uses to authenticate the user to the server when a connection is established. If set to **1 - Encrypt Password**, the driver sends the user ID in clear text and an encrypted password to the server for authentication. If set to **3 - Client Authentication**, the driver uses client authentication when establishing a connection. The database server relies on the client to authenticate the user and does not provide additional authentication. If set to **4 - Kerberos Authentication**, the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments. When set to **5 - Kerberos with UID & PWD**, the driver uses both Kerberos authentication and user ID and password authentication. The driver first authenticates the user using Kerberos. If a user ID and password are specified, the driver reauthenticates using the user name and password supplied. An error is generated if a user ID and password are not specified. If set to **6 - NTLM**, the driver uses NTLMv1 authentication for Windows clients. If set to **11 - SSL**, the driver uses SSL certificate information to authenticate the client with the server when using Oracle Wallet. The User Name and Password options should **not** be specified. See "Oracle Wallet SSL Authentication" for additional requirements. If set to **12 - SSL with UID & Password**, the driver uses user ID/password and SSL authentication to connect with the server when using Oracle Wallet. See "Oracle Wallet SSL Authentication" for additional requirements. If set to **16 - Wallet UID & PWD**, the driver authenticates to the server using a user ID and password retrieved from Oracle Wallet. See "Oracle Wallet Password Store" for additional requirements. If set to **38 - EntraIDAccessToken**, the driver authenticates to the server using an Entra ID access token. This setting requires the Entra Access Token option to be specified. If an access token is not specified, the driver throws an exception. All communications with the service are encrypted using TLS/SSL encryption. **Default: 1 - Encrypt Password**

Connection Options: Security	Description
GSS Client Library on page 213	The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC). **Default:** `native`
Credentials Wallet Path on page 191	Specifies the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. When Authentication Method is set to **16 - Wallet UID & PWD**, the driver retrieves the database user name and password from this file. See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.
Credentials Wallet Entry on page 190	Specifies the string value used to identify database credential information stored in an Oracle Wallet. When Authentication Method is set to **16 - Wallet UID & PWD**, the driver retrieves the user ID and password associated with the specified value from the wallet and uses them to authenticate to the server. This value provides a method for the correct user ID and password to be retrieved when there are multiple pairs in a wallet. See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.
Encryption Method on page 206	The method the driver uses to encrypt data sent between the driver and the database server. If set to **0 - No Encryption**, data is not encrypted. If set to **1 - SSL Auto**, data is encrypted using the TLS/SSL protocols specified in the Crypto Protocol Version connection option. **Default: 0 - No Encryption**
Crypto Protocol Version on page 192	Specifies the cryptographic protocols to use when TLS/SSL is enabled, where the highest version supported by the server is used. If none of the specified protocols are supported by the database server, the connection fails and the driver returns an error. **Default:** `TLSv1.3,TLSv1.2`
Validate Server Certificate on page 257	If 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 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:** Enabled

Connection Options: Security	Description
Enable FIPS on page 200	Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled (`Encryption Method=1`). If disabled, the OpenSSL library uses cryptographic algorithms from the default provider. If enabled, the OpenSSL library uses cryptographic algorithms from the FIPS provider. **Default:** Disabled
Trust Store on page 254	Specifies either the path and file name of the truststore file or the contents of the TLS/SSL certificates to be used when SSL is enabled (`Encryption Method=1`) and server authentication is used. **Default:** None
Trust Store Password on page 255	Specifies the password that is used to access the truststore file when TLS/SSL is enabled (`EncryptionMethod=1`) and server authentication is used. **Default:** None
Key Store on page 218	The absolute path of the keystore file to be used when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/SSL client authentication is enabled on the database server. **Default:** None
Key Store Password on page 219	The password used to access the keystore file when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/SSL client authentication is enabled on the database server. **Default:** None
Key Password on page 217	The password used to access the individual keys in the keystore file when TLS/SSL is enabled (`Encryption Method=1`) and TLS/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. **Default:** None
Host Name In Certificate on page 214	A host name for certificate validation when TLS/SSL encryption is enabled (`EncryptionMethod=1`) and validation is enabled (Validate Server Certificate=1). **Default:** None

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Performance tab allows you to specify performance data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

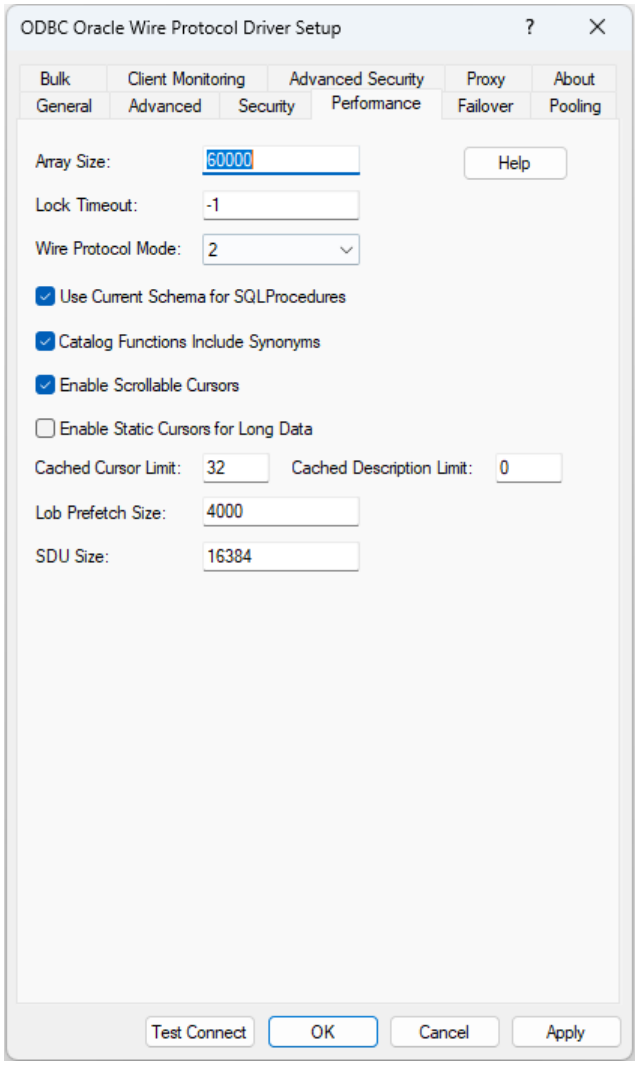
See also

Using security on page 119
Oracle Wallet Password Store on page 122
Oracle Wallet SSL Authentication on page 121
Data source configuration through a GUI on page 48

Performance tab

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

Figure 4: Performance tab



Connection Options: Performance	Description
Array Size on page 175	The number of bytes the driver can fetch in a single network round trip. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data. **Default:** `60000`

Connection Options: Performance	Description
Lock Timeout on page 227	Specifies the amount of time, in seconds, the Oracle server waits for a lock to be released before generating an error when processing a Select...For Update statement. If set to `-1`, the server waits indefinitely for the lock to be released. If set to `0`, the server generates an error immediately and does not wait for the lock to time out. If set to *x*, the server waits for the specified number of seconds for the lock to be released. **Default:** `-1`
Wire Protocol Mode on page 258	Specifies whether the driver optimizes network traffic to the Oracle server. If set to `1`, the driver operates in normal wire protocol mode without optimizing network traffic. If set to `2`, the driver optimizes network traffic to the Oracle server for result sets that contain repeating data in some or all of the columns, and the repeating data is in consecutive rows. It also optimizes network traffic if the application is updating or inserting images, pictures, or long text or binary data. **Default:** `2`
Use Current Schema for SQLProcedures on page 255	When enabled, the call for SQLProcedures is optimized, but only procedures owned by the current user are returned. When disabled, the driver does not limit the procedures returned. **Default:** Enabled
Catalog Functions Include Synonyms on page 183	Determines whether synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns. If enabled, synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns. If disabled, synonyms are excluded (a non-standard behavior) and performance is thereby improved. **Default:** Enabled
Enable Scrollable Cursors on page 202	Determines whether scrollable cursors, both Keyset and Static, are enabled for the data source. If set to enabled, scrollable cursors are enabled for the data source. If set to disabled, scrollable cursors are not enabled. **Default:** Enabled

Connection Options: Performance	Description
Enable Static Cursors for Long Data on page 204	Determines whether the driver supports Long columns when using a static cursor. Enabling this option causes a performance penalty at the time of execution when reading Long data. If enabled, the driver supports Long columns when using a static cursor. If disable, the driver does not support Long columns when using a static cursor. **Default:** Disabled
Cached Cursor Limit on page 182	Specifies the number of Oracle Cursor Identifiers that the driver stores in cache. A Cursor Identifier is needed for each concurrent open Select statement. **Default:** `32`
Cached Description Limit on page 182	Specifies the number of descriptions that the driver saves for Select statements. These descriptions include the number of columns, data type, length, and scale for each column. The matching is done by an exact-text match through the FROM clause. **Default:** `0`
LOB Prefetch Size on page 226	Specifies the size of prefetch data the server returns for BLOBs and CLOBs. LOB Prefetch Size is supported for Oracle database versions 12.1.0.1 and higher. If set to `-1`, the property is disabled. If set to `0`, the server returns only LOB meta-data such as LOB length and chunk size with the LOB locator during a fetch operation. If set to *x*, the server returns LOB meta-data and the beginning of LOB data with the LOB locator during a fetch operation. This can have significant performance impact, especially for small LOBs which can potentially be entirely prefetched, because the data is available without having to go through the LOB protocol. **Default:** `4000`
SDU Size on page 245	Specifies the size in bytes of the Session Data Unit (SDU) that the driver requests when connecting to the server. The SDU size is equivalent to the maximum number of bytes in a database protocol packets sent across the network. The setting of this option serves only as a suggestion to the database server. The actual SDU is negotiated with the database server. **Default:** `16384`

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally,you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.

- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

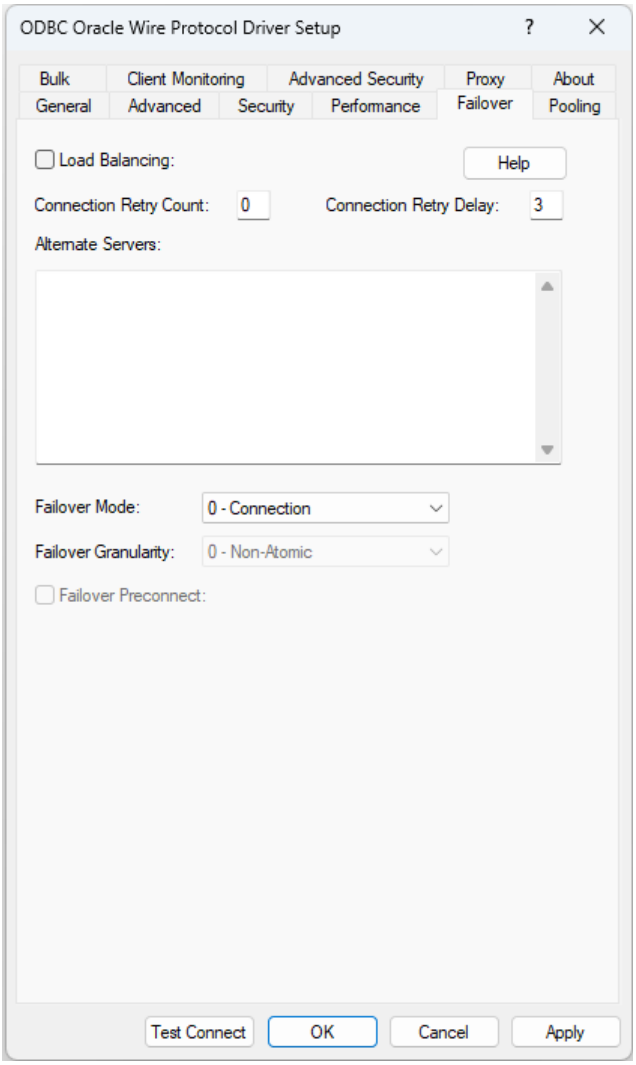
See also

Data source configuration through a GUI on page 48

Failover tab

The Failover tab allows you to specify your failover data source settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted. See "Using failover" for a general description of failover and its related connection options.

Figure 5: Failover tab



Connection Options: Failover	Description
Load Balancing on page 224	Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). If enabled, the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order. If 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). **Default:** Disabled

Connection Options: Failover	Description
Connection Retry Count on page 188	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. **Default:** `0`
Connection Retry Delay on page 189	Specifies the number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer. 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
Alternate Servers on page 173	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. For additional information, see "Alternate servers". **Default:** None
Failover Mode on page 210	Specifies the type of failover method the driver uses. If set to **0 - Connection**, the driver provides failover protection for new connections only. If set to **1 - Extended Connection**, the driver provides failover protection for new and lost connections, but not any work in progress. If set to **2 - Select**, the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed.

Connection Options: Failover	Description
Failover Granularity on page 209	Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection. 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**
Failover Preconnect on page 210	Specifies whether the driver tries to connect to the primary and an alternate server at the same time. If disabled, the driver tries to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection. If 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:** Disabled

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

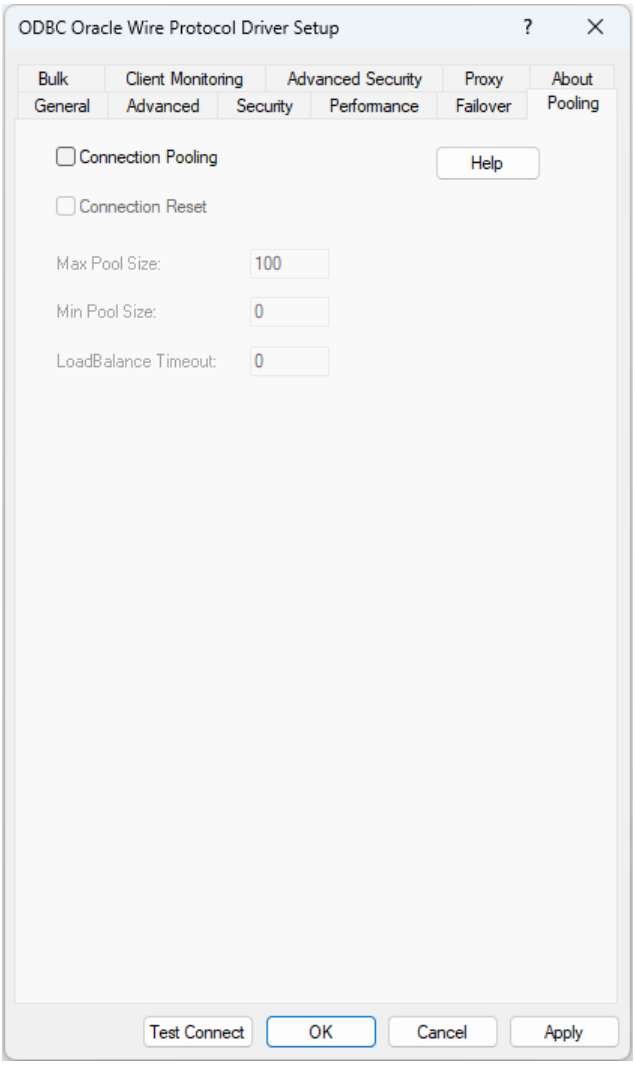
See also

Using failover on page 107
Alternate Servers on page 173
Data source configuration through a GUI on page 48

Pooling tab

The Pooling tab allows you to specify your pooling data source settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted. See "Using DataDirect Connection Pooling" for a general description of connection pooling.

Figure 6: Pooling tab



Connection Options: Pooling	Description
Connection Pooling on page 187	Specifies whether to use the driver's connection pooling. If enabled, the driver uses connection pooling. If disabled, the driver does not use connection pooling. **Default:** Disabled
Connection Reset on page 187	Determines whether the state of connections that are removed from the connection pool for reuse by the application is reset to the initial configuration of the connection. If 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 disabled, the state of connections is not reset. **Default:** Disabled
Max Pool Size on page 229	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. **Default:** `100`
Min Pool Size on page 229	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. **Default:** `0` (no connections are opened in addition to the current existing connection.)
LoadBalance Timeout on page 225	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. **Default:** `0` (inactive connections are kept open.)

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally,you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.

- Failover tab allows you to specify failover data source settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

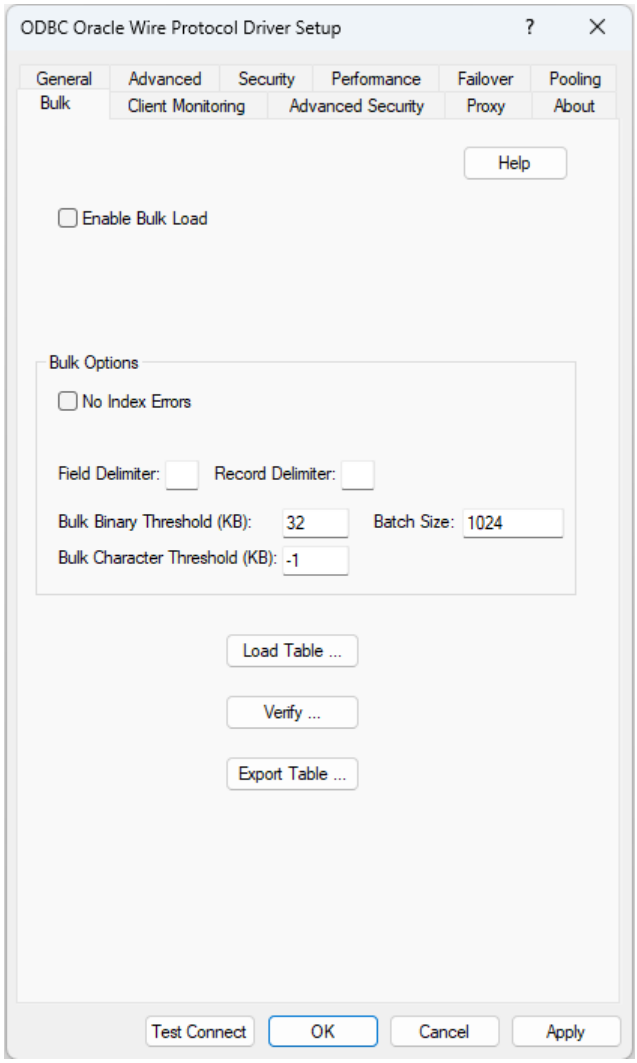
See also

Using DataDirect Connection Pooling on page 141
Data source configuration through a GUI on page 48

Bulk tab

The Bulk tab allows you to specify DataDirect Bulk Load data source settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted. See "Using DataDirect Bulk Load" for more information.

Figure 7: Bulk tab



Connection Options: Bulk	Description
Enable Bulk Load on page 199	Specifies the bulk load method. If 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 disabled, the driver uses standard parameter arrays. **Default:** Disabled
No Index Errors	Toggles options for the bulk load process. If enabled, the driver stops a bulk load operation when a value that would cause an index to be invalidated is loaded. For example, if a value is loaded that violates a unique or non-null constraint, the driver stops the bulk load operation and discards all data being loaded, including any data that was loaded prior to the problem value. If disabled, the bulk load operation continues even if a value that would cause an index to be invalidated is loaded. **Default:** Disabled
Field Delimiter on page 212	Specifies the character that the driver will use to delimit the field entries in a bulk load data file. **Default:** `,` (comma)
Record Delimiter on page 242	Specifies the character that the driver will use to delimit the record entries in a bulk load data file. **Default:** None
Bulk Binary Threshold on page 179	The maximum size, in KB, of binary data that is exported to the bulk data file. If set to `-1`, all binary data, regardless of size, is written to the bulk data file, not to an external file. If set to `0`, all binary data, regardless of size, is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file. If set to *x*, any binary data exceeding this specified number of KB is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file. **Default:** None

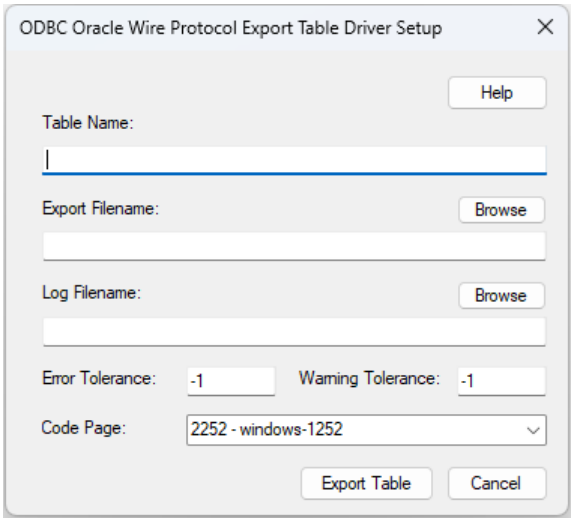
Connection Options: Bulk	Description
Batch Size on page 177	The number of rows that the driver sends to the database at a time during bulk operations. This value applies to all methods of bulk loading. Default: `1024`
Bulk Character Threshold on page 180	The maximum size, in KB, of character data that is exported to the bulk data file. 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`

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

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

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

   Figure 8: Export Table dialog box

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

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

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

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. The file name must be the fully qualified path to the log file. Events logged to this file are:

- Total number of rows fetched
- A message for each row that failed to export
- Total number of rows that failed to export
- Total number of rows successfully exported

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

If you do not 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" 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**.

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

2. 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" for details. The ODBC Oracle Wire Protocol Verify Driver Setup dialog box appears.

Figure 9: ODBC Oracle Wire Protocol Verify Driver Setup dialog box

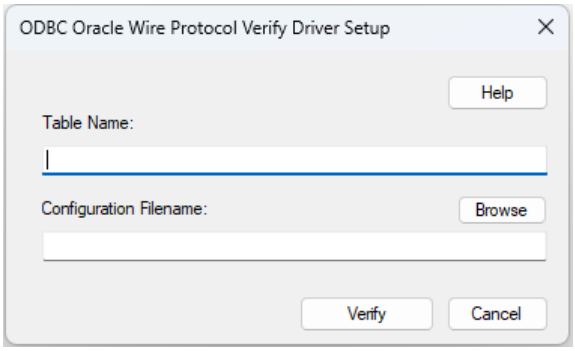


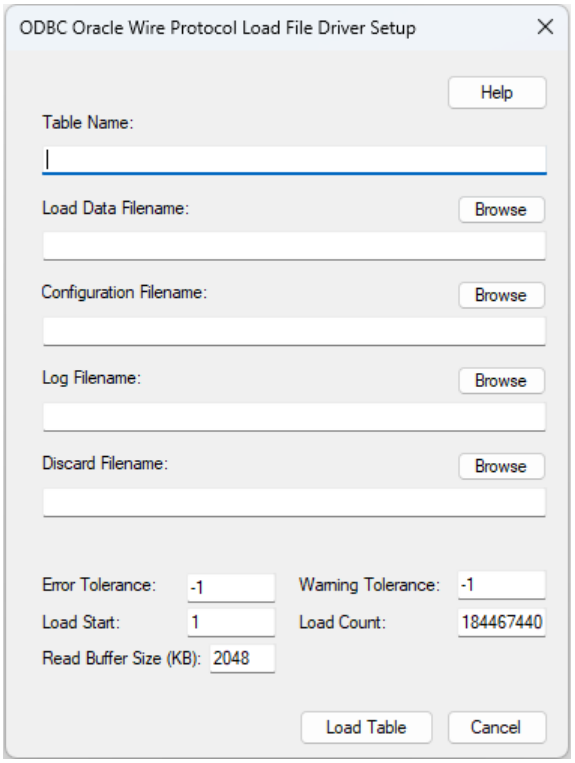
Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file. The file name must be the fully qualified path to the configuration file.

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

3. 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 10: Load File dialog box



The load operation can create a log file and can also create a discard file that contains rows rejected during the load. The discard file is in the same format as the bulk load data file. After fixing reported issues in the discard file, the bulk load can be reissued using the discard file as the bulk load data file.

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

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

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

Table Name: A string that specifies the name of the target database table into which the data is loaded.

Load Data Filename: A string that specifies the path (relative or absolute) and file name of the bulk data file from which the data is loaded. The file name must be the fully qualified path to the bulk data file.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file. The file name must be the fully qualified path to the configuration file.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The file name must be the fully qualified path to the log file. Specifying a value for Log Filename creates the file if it does not already exist. Events logged to this file are:

- Total number of rows read
- Message for each row that failed to load
- Total number of rows that failed to load
- Total number of rows successfully loaded

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

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

Discard Filename: A string that specifies the path (relative or absolute) and file name of the bulk discard file. The file name must be the fully qualified path to the discard file. Any row that cannot be inserted into database as result of bulk load is added to this file, with the last row rejected added to the end of the file.

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

If you do not specify a value for Discard Filename, a discard file is not created.

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

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

Load Start: A value that specifies the first row to be loaded from the data file. Rows are numbered starting with 1. For example, when Load Start is 10, the first 9 rows of the file are skipped and the first row loaded is row 10. This option can be used to restart a load after a failure.

The default value is `1`.

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

The default value is `2048`.

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

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

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

The default value is the maximum value for SQLULEN. If set to `0`, no rows are loaded.

Click **Load Table** to connect to the database and load the table or click **Cancel**.

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally,you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Client Monitoring tab allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

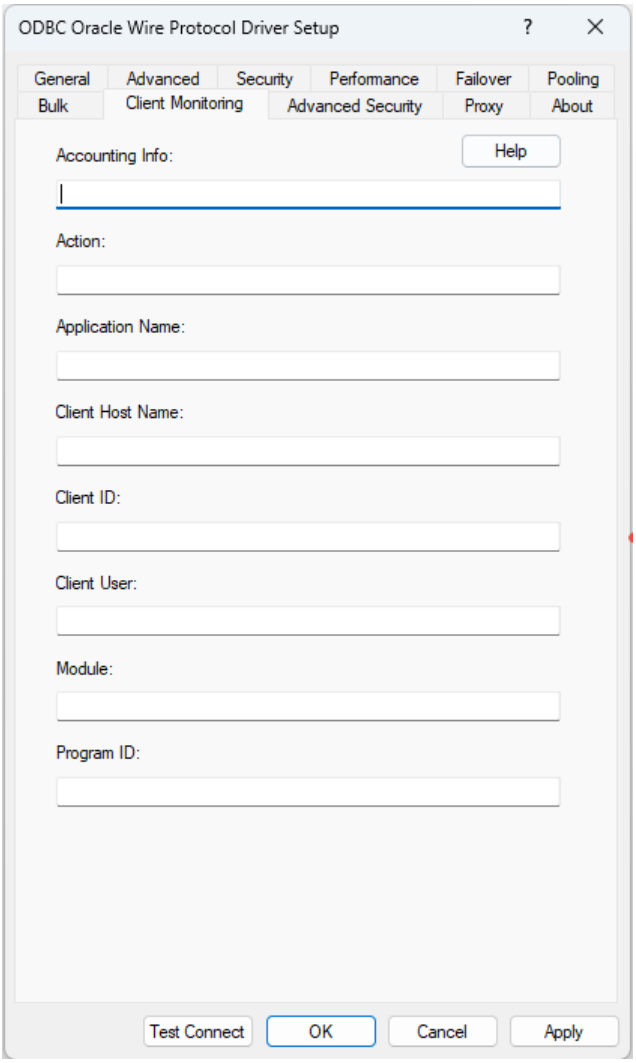
See also

Using DataDirect Bulk Load on page 145
External overflow files on page 152
Character set conversions on page 152
Verification of the bulk load configuration file on page 150
Data source configuration through a GUI on page 48

Client Monitoring tab

The Client Monitoring tab allows you to specify additional data source settings. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted. See "Using client information" for more information.

Figure 11: Client Monitoring tab



Connection Options: Client Monitoring	Description
Accounting Info on page 172	Accounting information to be stored in the database. This value sets the CLIENT_INFO value of the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Action on page 172	The current action (Select, Insert, Update, or Delete, for example) within the current module. This value sets the ACTION column of the V$SESSION table on the server. This value is used by the client information feature. **Default:** None

Connection Options: Client Monitoring	Description
Application Name on page 174	The name of the application to be stored in the database. This value sets the dbms_session value in the database and the PROGRAM value of the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Client Host Name on page 184	The host name of the client machine to be stored in the database. This value sets the MACHINE value in the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Client ID on page 185	Additional information about the client to be stored in the database. This value sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Client User on page 186	The user ID to be stored in the database. This value sets the OSUSER value in the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Module on page 230	Provides additional information about the client to be stored in the database. This value sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server. This value is used by the client information feature. **Default:** None
Program ID on page 241	The product and version information of the driver on the client to be stored in the database. This value sets the PROCESS value in the V$SESSION table on the server. This value is used by the client information feature. **Default:** None

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally,you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.

- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).
- Proxy tab allows you to specify settings for connecting through an HTTP proxy.

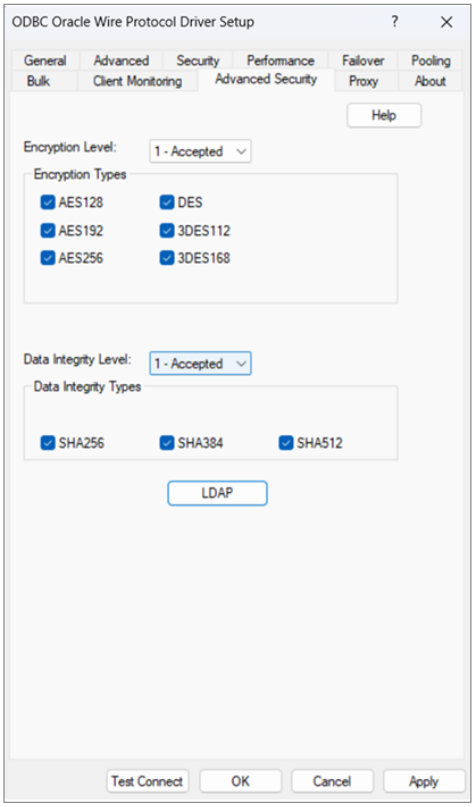
See also

Using client information on page 117
Data source configuration through a GUI on page 48

Advanced Security tab

The Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS). On this tab, provide values for the options in the following tables; then, click **Apply**. The fields are optional unless otherwise noted.

Figure 12: Advanced Security tab

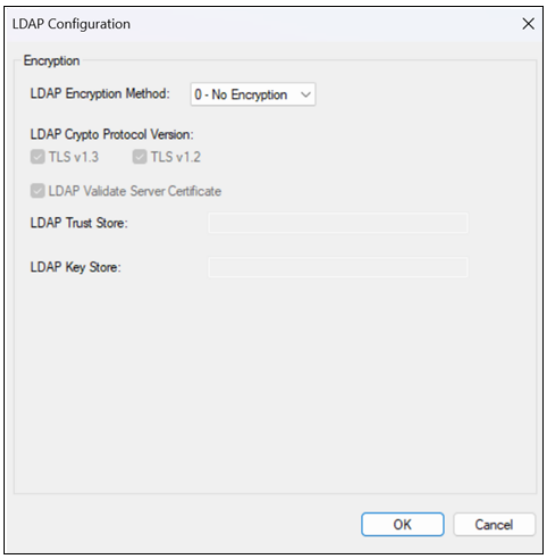


Connection Options: Advanced Security	Description
Encryption Level on page 205	Specifies a preference on whether to use encryption on data being sent between the driver and the database server. If set to **0 - Rejected**, or if no match is found between the driver and server encryption types, data sent between the driver and the database server is not encrypted or decrypted. The connection fails if the database server specifies REQUIRED. If set to **1 - Accepted**, encryption is used on data sent between the driver and the database server if the database server requests or requires it. If set to **2 - Requested**, data sent between the driver and the database server is encrypted and decrypted if the database server permits it. If set to **3 - Required**, data sent between the driver and the database server must be encrypted and decrypted. The connection fails if the database server specifies REJECTED. **Default: 1 - Accepted**
Encryption Types on page 208	Specifies the encryption algorithms to use if Oracle Advanced Security encryption is enabled using the Encryption Level connection property. **Default:** All listed encryption algorithms are selected.

Connection Options: Advanced Security	Description
Data Integrity Level on page 194	Specifies a preference for the data integrity to be used on data sent between the driver and the database server. The connection fails if the database server does not have a compatible integrity algorithm. If set to **0 - Rejected**, a data integrity check on data sent between the driver and the database server is refused. The connection fails if the database server specifies REQUIRED. If set to **1 - Accepted**, a data integrity check can be made on data sent between the driver and the database server. Data integrity is used if the database server requests or requires it. If set to **2 - Requested**, the driver enables a data integrity check on data sent between the driver and the database server if the database server permits it. If set to **3 - Required**, a data integrity check must be performed on data sent between the driver and the database server. The connection fails if the database server specifies REJECTED. See "Encryption and Data Integrity" for more information. **Default: 1 - Accepted**
Data Integrity Types on page 195	Determines the method the driver uses to protect against attacks that intercept and modify data being transmitted between the client and server. You can enable data integrity protection without enabling encryption. If multiple values are specified and Oracle Advanced Security data integrity is enabled using the Data Integrity Level option, the database server determines which algorithm is used based on how it is configured. **Default: MD5**, **SHA1**, **SHA256**, **SHA384**, **SHA512**

On the Advanced Security tab, you can configure LDAP settings as well. To specify values for the LDAP options, click **LDAP**. The LDAP Configuration dialog box appears.

Figure 13: LDAP Configuration tab



Connection Options: LDAP Configuration	Description
LDAP Encryption Method on page 221	Determines whether data is encrypted and decrypted when transmitted over the network between the driver and LDAP server. If set to **0 - No Encryption**, data is neither encrypted nor decrypted. If set to **1 - SSL**, data is encrypted using TLS/SSL. If the LDAP server does not support TLS/SSL, the connection fails and the driver throws an exception. **Default: 0 - No Encryption**
LDAP Crypto Protocol Version on page 219	Specifies a cryptographic protocol or comma-separated list of cryptographic protocols that can be used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). **Default:** `TLSv1.3,TLSv1.2`
LDAP Validate Server Certificate on page 224	Determines whether the driver validates the certificate sent by the server when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). **Default:**`1`

Connection Options: LDAP Configuration	Description
LDAP Trust Store on page 223	Specifies the absolute path to the truststore file used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). **Default:** No default value
LDAP Key Store on page 222	Specifies the absolute path to the keystore file used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). **Default:** No default value

If you have finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab allows you to specify additional data source settings.

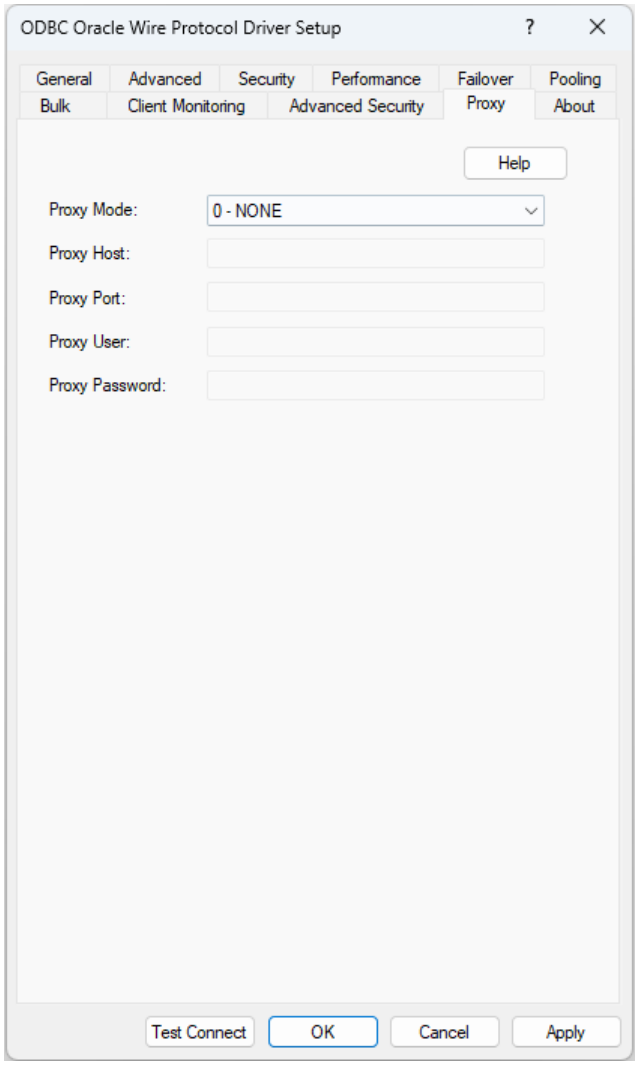
See also

Oracle Advanced Security on page 135
Data source configuration through a GUI on page 48

Proxy tab

The Proxy tab allows you to specify settings for connecting through an HTTP proxy. On this tab, provide values for the options in the following table; then, click **Apply**. The fields are optional unless otherwise noted.

Figure 14: Proxy tab



Connection Options: Advanced Security	Description
Proxy Mode on page 234	Determines whether the driver connects to an endpoint through an HTTP proxy server. If set to **0 - NONE**, the driver connects directly to the endpoint specified by the Host Name connection option. If set to **1 - HTTP**, the driver connects to the endpoint through the HTTP proxy server specified by the ProxyHost connection option. **Default: 0 - None**

Connection Options: Advanced Security	Description
Proxy Host on page 234	Specifies the Hostname and possibly the Domain of the Proxy Server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address. **Default:** Empty string
Proxy Port on page 236	Specifies the port number where the Proxy Server is listening for HTTP requests. **Default:** `0`
Proxy User on page 237	Specifies the user name needed to connect to the Proxy Server. **Default:** Empty string
Proxy Password on page 235	Specifies the password needed to connect to the Proxy Server. **Default:** Empty string

If you finished configuring your driver, proceed to Step 6 on page 52 in "Data source configuration through a GUI." Optionally, you can further configure your driver by clicking on the following tabs. The following sections provide details on the fields specific to each configuration tab:

- General tab allows you to configure options that are required for creating a data source.
- Advanced tab allows you to configure advanced behavior.
- Security tab allows you to specify security data source settings.
- Performance tab allows you to specify performance data source settings.
- Failover tab allows you to specify failover data source settings.
- Pooling tab allows you to specify connection pooling settings.
- Bulk tab allows you to specify data source settings for DataDirect Bulk Load.
- Client Monitoring tab allows you to specify additional data source settings.
- Advanced Security tab allows you to specify settings for Oracle Advanced Security (OAS).

See also

Connecting through a proxy server on page 99
Data source configuration through a GUI on page 48

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]...]
```

For connection strings using a DSN or File DSN, this means that all required connection information must be provided in the data source and/or connection string.

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 "Connection Option Descriptions" section 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 Oracle Wire Protocol is:

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

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

```
FILEDSN=OracleWP.dsn;ID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;HOST=server1;PORT=1522;
UID=JOHN;PWD=XYZZY;SERVICENAME=SALES.US.ACME.COM
```

See also

Connection option descriptions on page 163

Password Encryption Tool (UNIX/Linux only)

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

- KeyPassword
- KeyStorePassword
- TrustStorePassword
- Password

To use the Password Encryption Tool:

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

   `ddencpwd` *`password`*

   where:

   `password`

   is the password you want to encrypt.

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

   `KeyPassword=`*`returned_value`*

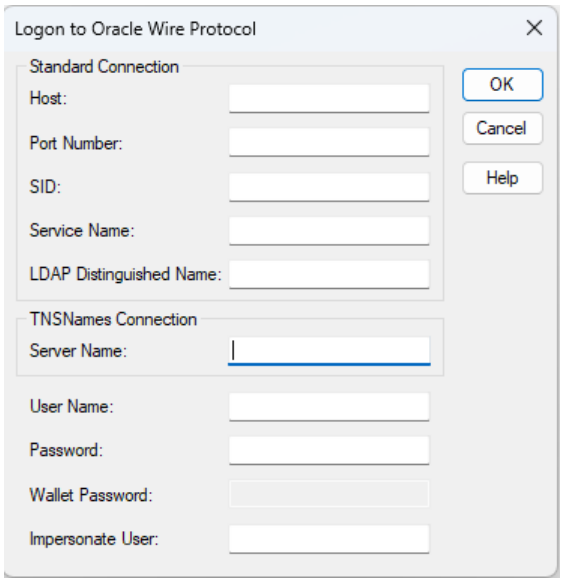
4. Repeat Steps 2 and 3 to encrypt additional passwords.
5. If using an `odbc.ini` file, save your file.

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

Using a logon dialog box

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

Figure 15: Logon to Oracle Wire Protocol dialog box



In this dialog box, provide the following information:

Note: For TNSNames connections, skip to Step 4 on page 91.

1. In the Host field, type either the name or the IP address of the server to which you want to connect. Note:

   Note:

   - The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.
   - If you enter a value for this field, the Server Name field is not available.
   - If you enter a value for the LDAP Distinguished Name field, this field specifies name or address of the LDAP directory server.
   - This field is not available if you enter a value for the Server Name field.

2. In the Port Number field, type the number of your Oracle listener. Check with your database administrator for the correct number.

   Note:

   - If you enter a value for this field, the Server Name field is not available.
   - If you enter a value for the LDAP Distinguished Name field, this field specifies the port number listener of the LDAP directory server.
   - This field is not available if you enter a value for the Server Name field.

3. Provide a value for one of the following options:

   - In the SID field, type the Oracle System Identifier that refers to the instance of Oracle running on the server.
   - In the Service Name field, type the Oracle service name that specifies the database used for the connection. See "Service Name" in "Connection option descriptions" for details.
   - In the LDAP Distinguished Name field, type the fully qualified path of names in the LDAP directory information tree for the entry containing your connection information.

   Skip to Step 6 on page 91

4. In the Server Name field, type a net service name that exists in the `TNSNAMES.ORA` file. The corresponding entry in the `TNSNAMES.ORA` file is used to obtain Host, Port Number, and SID information.

   Note:

   - If you enter a value for this field, the Host, Port Number, SID, and Service Name fields are not available.
   - If you enter a value for either the Host, Port Number, SID, or Service Name fields, this field is not available.

5. If you are using an Oracle Wallet Password Store (`AuthenticationMethod=16`), in the Oracle Wallet field, type your the password used to access the Oracle Wallet in which your database credential information is stored. Skip to Step 7 on page 92
6. If required, provide values for one of the following fields:

- Password: Type your Oracle user name.
- Entra Access Token: Enter your Entra ID Access Token.

Note: Entra Access Token is only exposed when using Entra ID access token authentication (`AuthenticationMethod=38`)

7. Optionally, in the Impersonate User field, type the proxy user ID used for impersonation. This value determines your identity and permissions when executing queries. Note that the administrator must grant permission to the authenticated user to impersonate the specified proxy user ID.
8. Click **OK** to log on to the Oracle database installed on the server you specified and to update the values in the Registry.

Note: You can also use OS Authentication to connect to an Oracle database. See "OS authentication" for details.

See also

Connection option descriptions on page 163
OS authentication on page 104
Service Name on page 247

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.

Array Size (ArraySize): If this connection string attribute is set appropriately, the driver can improve performance of your application by reducing the number of round trips on the network. For example, if your application normally retrieves 200 rows, it is more efficient for the driver to retrieve 200 rows at one time over the network than to retrieve 50 rows at a time during four round trips over the network.

Cached Cursor Limit (CachedCursorLimit): To improve performance when your application executes concurrent Select statements, Cursor Identifiers can be cached. In this case, the Cursor Identifier is retrieved from a cache rather than being created for each connection. When an Identifier is needed, the driver takes one from its cache, if one is available, rather than creating a new one. Cached Cursor Identifiers are closed when the connection is closed. To cache Cursor Identifiers, the CachedCursorLimit attribute must be set to the appropriate number of concurrent open Select statements.

Cached Description Limit (CachedDescLimit): The driver can cache descriptions of Select statements and improve the performance of your ODBC application; therefore, if your application issues a fixed set of SQL queries throughout the life of the application, the description of the query should be cached. If a description is not cached, the description must be retrieved from the server, which reduces performance. The descriptions include the number of columns and the data type, length, and scale for each column. The matching is done by an exact-text match through the From clause. If the statement contains a Union or a subquery, the driver cannot cache the description.

Catalog Functions Include Synonyms (CatalogIncludesSynonyms): Standard ODBC behavior is to include synonyms in the result set of calls to the following catalog functions: SQLProcedures, SQLStatistics and SQLProcedureColumns. Retrieving this synonym information degrades performance. If your ODBC application does not need to return synonyms when using these catalog functions, the driver can improve performance if the CatalogIncludesSynonyms attribute is disabled (set to `0`).

Catalog Options (CatalogOptions): If your application does not need to access the comments/remarks for database tables, performance of your application can be improved. In this case, the CatalogOptions attribute should be disabled (set to `0`) because retrieving comments/remarks degrades performance. If this attribute is enabled (set to `1`), result column REMARKS (for the catalog functions SQLTables and SQLColumns) and the result column COLUMN_DEF (for the catalog function SQLColumns) return actual values.

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

- **Accounting Info (AccountingInfo)**: Sets the CLIENT_INFO value of the V$SESSION table on the server.
- **Action (Action)**: Sets ACTION column of the V$SESSION table on the server.
- **Application Name (ApplicationName)**: Sets the dbms_session value in the database and the PROGRAM value of the V$SESSION table on the server.
- **Client Host Name (ClientHostName)**: Sets the MACHINE value in the V$SESSION table on the server.
- **Client ID (ClientID)**: Sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server.
- **Client User (ClientUser)**: Sets the OSUSER value in the V$SESSION table on the server.
- **Module (Module)**: Sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server.
- **Program ID (ProgramID)**: Sets the PROCESS value in the V$SESSION table on the server.

Connection Pooling (Pooling): On Windows, UNIX, or Linux, 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.

Data Integrity Level (DataIntegrityLevel) and **Data Integrity Types (DataIntegrityTypes)**: Checking data integrity may adversely reduce performance because of the additional overhead (mainly CPU usage) that is required to perform the check.

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.

Describe At Prepare (DescribeAtPrepare): When enabled, this option requires extra network traffic. If your application does not require result set information at prepare time (for instance, you request information about the result set using SQLColAttribute(s), SQLDescribeCol, SQLNumResultCols, and so forth, before calling SQLExecute on a prepared statement), you can increase performance by disabling this option.

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.

EnableServerResultCache (EnableServerResultCache): If your application connects to Oracle 11*g* and executes the same query multiple times, you can improve performance by using the Oracle feature server-side resultset caching. When enabled, Oracle stores the result set in database memory. On subsequent executions of the same query, the result set is returned from database memory if the underlying tables have not been modified. Without result set caching, the server would process the query and formulate a new result set.

Enable Scrollable Cursors (EnableScrollableCursors) and **Enable Static Cursors for Long Data (EnableStaticCursorsForLongData)**: When your application uses Static or Keyset (Scrollable) cursors, the EnableScrollableCursors attribute must be enabled (set to `1`). Also, if your application retrieves images, pictures, long text or binary data while using Static cursors, the EnableStaticCursorsForLongData attribute must be enabled (set to `1`). However, this can degrade performance when retrieving long data with Static cursors as the entire result set is stored on the client. To improve performance, you might consider designing your application to retrieve long data through forward-only cursors.

Encryption Method (EncryptionMethod), Encryption Level (EncryptionLevel), and **Encryption Types (EncryptionTypes)**: Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) required to encrypt and decrypt data. Using data encryption can degrade performance more than performing data integrity checks.

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.

LOB Prefetch Size (LOBPrefetchSize): You can improve performance when fetching LOBs by enabling the LOB Prefetch Size option (set to value equal to or greater than `0`). With LOB prefetching enabled, the driver can return LOB meta-data and the beginning of LOB data along with the LOB locator during a fetch operation, therefore reducing the number of round trips and improving performance. For significant gains, specify a value that is large enough to entirely prefetch LOB values. This allows data to be available without having to go through LOB protocol, which can be expensive.

Lock Timeout (LockTimeOut): Sometimes users attempt to select data that is locked by another user. Oracle provides three options when accessing locked data with SELECT … FOR UPDATE statements:

- Wait indefinitely for the lock to be released (`-1`)
- Return an error immediately (`0`)
- Return an error if the lock has not been released within a specific number of seconds (*n* seconds)

Some applications may benefit by not waiting indefinitely and continuing execution; this keeps the application from hanging. The application, however, needs to handle lock timeouts properly with an appropriate timeout value; otherwise, processing time could be wasted handling lock timeouts, and deadlocks could go undetected.

To improve performance, either enter a number of seconds or enter `0` as the value for this option.

Procedure Returns Results (ProcedureRetResults): The driver can be tuned for improved performance if your application's stored procedures do not return results. In this case, the ProcedureRetResults attribute should be disabled (set to `0`).

SDU Size (SDUSize): Set this option based on the size of result sets returned by your application. If your application returns large result sets, set this option to the maximum SDU size configured on the database server. This reduces the total number of packets required to return data to the client, thus improving performance. If your application returns small result sets, set this option to a size smaller than the maximum to avoid burdening your network with unnecessarily large packets.

Server Process Type (ServerType): When using a dedicated server connection, a server process on UNIX (a thread on Windows) is created to serve only your application connection. When you disconnect, the process goes away. The socket connection is made directly between your application and this dedicated server process. This can provide tremendous performance improvements, but will use significantly more resources on UNIX servers. Because this is a thread on Oracle servers running on Windows platforms, the additional resource usage on the server is significantly less. This option should be set to `2` (dedicated) when you have a batch environment with lower numbers of connections, your Oracle server has excess processing capacity and memory available when at maximum load, or if you have a performance-sensitive application that would be degraded by sharing Oracle resources with other applications.

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

Using LDAP

LDAP (Lightweight Directory Access Protocol) is a protocol used to access and manage distributed directory information services over a network.

An LDAP directory is a specialized database specifically designed for searching and retrieving data. In an LDAP directory, data is organized into objects called entries. Each entry has a unique Distinguished Name and contains a set of attributes that define the characteristics of the object the entry represents (like a user, group, or device). The entries are arranged in a hierarchical tree-like structure.

The LDAP directory is hosted on an LDAP server, which provides the LDAP service by responding to LDAP protocol requests.

The driver supports connecting to the LDAP server using the following connection methods. After establishing a connection to the LDAP server, the driver retrieves the necessary connection details from it and then connects to the database server.

- Unencrypted connection
- Encrypted connection: TLS/SSL encryption for server validation
- Encrypted connection: TLS/SSL encryption for server and client validation

Note: In the procedures and examples provided, the driver uses user ID/password authentication for authenticating to the database server, but you can use any of the supported authentications. For details, see "Authentication".

Unencrypted connection

To connect to the LDAP server using an unencrypted connection:

- Set the LDAP Distinguished Name (LDAPDistinguishedName) connection option to specify the fully qualified path of names in the LDAP directory information tree for the entry containing your connection information. For example:

  ```
  cn=DB122,cn=OracleContext,dc=america,dc=yourcompany,dc=com
  ```

- Set the Host (HostName) connection option to specify the name or IP address of the LDAP server.
- Set the Port Number (PortNumber) connection option to specify the port number listener of the LDAP server. The default value is 389.
- Set the User connection option to specify your user name for the database server.
- Set the Password connection option to specify the password for the database server.

The following connection string and `odbc.ini` file examples show how to configure the driver for an unencrypted connection to the LDAP server:

Connection string

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;HostName=ldap.company.com;
       PortNumber=389;LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com;

       user=dbuser;password=dbpass;
```

odbc.ini

```
[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
...
Description=DataDirect 8.0 Oracle Wire Protocol
...
HostName=ldap.company.com
...
PortNumber=389
...
LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com
...
user=dbuser
...
password=dbpass
...
```

Encrypted connection: TLS/SSL encryption for server validation

To connect to the LDAP server using TLS/SSL encryption for server validation:

- Set the LDAP Encryption Method (LDAPEncryptionMethod) connection option to 1.
- Set the LDAP Distinguished Name (LDAPDistinguishedName) connection option to specify the fully qualified path of names in the LDAP directory information tree for the entry containing your connection information. For example:

  ```
  cn=DB122,cn=OracleContext,dc=america,dc=yourcompany,dc=com
  ```

- Set the Host (HostName) connection option to specify the name or IP address of the LDAP server.
- Set the Port Number (PortNumber) connection option to specify the port number listener of the LDAP server. The default value is 636.
- Set the LDAP Trust Store (LDAPTrustStore) connection option to specify the absolute path to the truststore file that contains certificates that the client uses to verify the LDAP server's certificate.
- Set the LDAP Validate Server Certificate (LDAPValidateServerCertificate) connection option to 1.
- Set the User connection option to specify your user name for the database server.
- Set the Password connection option to specify the password for the database server.

The following connection string and odbc.ini file examples show how to configure the driver for an encrypted connection to the LDAP server using TLS/SSL encryption for server validation:

Connection string

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;
       HostName=ldap.company.com;PortNumber=636;LDAPEncryptionMethod=1;
       LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com;
       LDAPTrustStore=C:/certs/truststore.crt;LDAPValidateServerCertificate=1;
       user=dbuser;password=dbpass
```

odbc.ini

```
[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
...
Description=DataDirect 8.0 Oracle Wire Protocol
...
HostName=ldap.company.com
...
PortNumber=636
...
LDAPEncryptionMethod=1
...
LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com
...
LDAPTrustStore=C:/certs/truststore.crt
...
LDAPValidateServerCertificate=1
...
user=dbuser
...
password=dbpass
...
```

Encrypted connection: TLS/SSL encryption for both server and client validation

To connect to the LDAP server using TLS/SSL encryption for both server and client validation:

- Set the LDAP Encryption Method (LDAPEncryptionMethod) connection option to 1.
- Set the LDAP Distinguished Name (LDAPDistinguishedName) connection option to specify the fully qualified path of names in the LDAP directory information tree for the entry containing your connection information. For example:

  ```
  cn=DB122,cn=OracleContext,dc=america,dc=yourcompany,dc=com
  ```

- Set the Host (HostName) connection option to specify the name or IP address of the LDAP server.
- Set the Port Number (PortNumber) connection option to specify the port number listener of the LDAP server. The default value is 636.

- Set the LDAP Trust Store (LDAPTrustStore) connection option to specify the absolute path to the truststore file that contains certificates that the client uses to verify the LDAP server's certificate.

  Note: The driver supports only the `.pem` file format for the truststore files.

  The driver supports only the `.pem` file format for the keystore files.

- Set the LDAP Key Store (LDAPKeyStore) connection option to specify the absolute path to the keystore file that contains certificates that the client presents in response to the LDAP server's certificate request.

  Note: The driver supports only the `.pem` file format for the keystore files.

- Set the LDAP Validate Server Certificate (LDAPValidateServerCertificate) connection option to `1`.
- Optionally, set the LDAP Crypto Protocol Version (LDAPCryptoProtocolVersion) connection option to specify the cryptographic protocol you want to use for TLS/SSL encryption. If not specified, the cryptographic protocol used depends on the highest protocol version supported by the server. The driver supports TLSv1.3 and TLSv1.2.
- Set the User connection option to specify your user name for the database server.
- Set the Password connection option to specify the password for the database server.

The following example connection string shows how to configure the driver for an encrypted connection to the LDAP server using TLS/SSL encryption for both server and client validation:

Connection string

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;
       HostName=ldap.company.com;PortNumber=636;LDAPEncryptionMethod=1;
       LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com;
       LDAPTrustStore=C:/certs/truststore.pem;LDAPKeyStore=C:/certs/keystore.pem;
       LDAPValidateServerCertificate=1;user=dbuser;password=dbpass
```

odbc.ini

```
[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
...
Description=DataDirect 8.0 Oracle Wire Protocol
...
HostName=ldap.company.com
...
PortNumber=636
...
LDAPEncryptionMethod=1
...
LDAPDistinguishedName=cn=MYDB,cn=OracleContext,dc=company,dc=com
...
LDAPTrustStore=C:/certs/truststore.pem
...
LDAPKeyStore=C:/certs/keystore.pem
...
LDAPValidateServerCertificate=1
...
user=dbuser
...
password=dbpass
...
```

See also

LDAP Distinguished Name on page 220

Host on page 213
Port Number on page 233

Connecting through a proxy server

Supported on Windows, UNIX, and Linux only.

In some environments, your application may need to connect through an HTTP proxy, for example, if your application uses a web server or gateway system to access server clusters.

Note: Oracle Connection Manager is not currently supported using the procedure described in this section. See "Oracle Connection Manager" for more information.

To connect to a server through an HTTP proxy:

- Set the service name *or* SID:
  - Set the Service Name (`ServiceName`) option to specify the Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name, for example: `sales.us.acme.com`.
  - Set the SID (`SID`) option to specify the Oracle System Identifier that refers to the instance of Oracle running on the server.
- Set the Host (`HostName`) option to specify the name or the IP address of the database server to which you want to connect.
- Set the Port Number (`PortNumber`) option to specify the port number of the database server listener.
- Set the proxy server specific options:
  - Set the Proxy Mode (`ProxyMode`) option to `1` (HTTP).
  - Set the Proxy Host (`ProxyHost`) option to specify the Hostname and, if required by your environment, the Domain, of the proxy server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.
  - Set the Proxy Port (`ProxyPort`) option to specify the port number where the proxy server is listening for HTTP requests. The default is `0`.
  - Optionally, set the Proxy User (`ProxyUser`) option to specify the user name used to connect to the Proxy Server.
  - Optionally, set Proxy Password (`ProxyPassword`) to specified the password needed to connect to the proxy server.

The following examples demonstrate a basic connection to a proxy server.

Using a connection string:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;HostName=123.210.123.210;
PortNumber=5439;ProxyHost=123.255.78.90;ProxyMode=1;ProxyPort=1521;
ServiceName=sales.us.example.com;ProxyUser=jqpublic;ProxyPassword=secret;
```

Using the odbc.ini file with a 32-bit driver:

```
Driver=ODBCHOME/lib/ivoraXX.so
Description=DataDirect Oracle Wire Protocol driver
HostName=123.210.123.210
PortNumber=5439
ProxyHost=123.255.78.90
ProxyMode=1
ProxyPassword=secret
ProxyPort=1521
ProxyUser=jqpublic
ServiceName=sales.us.acme.com
```

See also

Oracle Connection Manager on page 100
Service Name on page 247
SID on page 248
Host on page 213
Port Number on page 233
Proxy Mode on page 234
Proxy Host on page 234
Proxy Port on page 236
Proxy User on page 237
Proxy Password on page 235

Oracle Connection Manager

Supported on Windows, UNIX, and Linux only.

Oracle Connection Manager is a network solution that serves as a proxy to Oracle database servers and clusters. In addition to being a single point of access, Oracle Connection Manager offers a number of network solutions, including increased scalability, simplified access control, and improved availability. The driver supports Oracle Connection Manager for connections that are defined using the TNSNAMES.ORA file.

To connect to Oracle Connection Manager:

- In the TNSNAMES.ORA file:
  - Define the net service name entry for your Oracle Connection Manager service. When using Oracle Connection Manager, the definition must contain the keyword-value pair SOURCE_ROUTE=YES. Refer to the documentation for your Oracle database for details and the latest information.

    Note: Oracle net service name definitions support the SOURCE_ROUTE keyword at the DESCRIPTION_LIST, DESCRIPTION, and ADDRESS_LIST levels. Currently, the driver supports defining SOURCE_ROUTE at the DESCRIPTION and ADDRESS_LIST levels, but not at the DESCRIPTION_LIST level.

- For the driver:

- Set the TNSNames File (`TNSNamesFile`) to specify the name(s) and location(s) of the `TNSNAMES.ORA` file(s) that contains the net service name definition for your Oracle Connection Manager service.
- Set the Server Name (`ServerName`) option to specify the net service name for the Oracle Connection Manger service defined in the `TNSNAMES.ORA` file. The corresponding net service name entry in the `TNSNAMES.ORA` file is used to obtain connection information.

The following examples demonstrate a basic connection through Oracle Connection Manager.

Using a connection string:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;
ServerName=MyNetServiceName;TNSNamesFile=F:/server2/oracle/tnsnames.ora;
```

Using the `odbc.ini` file with a 32-bit driver:

```
Driver=ODBCHOME/lib/ivoraXX.so
Description=DataDirect Oracle Wire Protocol driver
ServerName=MyNetServiceName
TNSNamesFile=F:/server2/oracle/tnsnames.ora
```

See also

Server Name on page 246
TNSNames File on page 252

Support for Oracle RAC

Oracle introduced Real Application Clusters (RAC) with Oracle 9*i*, and RAC continues to be a key feature for the current generation of databases. Oracle RAC allows a single physical Oracle database to be accessed by concurrent instances of Oracle running across several different CPUs.

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

Connection failover and *client load balancing* can be used in conjunction with an Oracle RAC system, but they are not specifically part of Oracle RAC.

See also

Using failover on page 107

XA interface support

The driver enables support for distributed transactions by implementing the XA interface. In an X/Open Distributed Transaction Processing (DTP) system, the XA interface provides a method for the Transaction Manager (TM) to call xa_ routines to interact with the Resource Manager (RM).

For more information on the X/Open DTP system, the XA interface, and the XA components discussed in the following procedure, refer to Distributed Transaction Processing: The XA Specification.

To perform distributed transactions:

1. Establish a connection between your application and the RM (the database server you want to connect to) in AUTOCOMMIT_OFF mode. For example:

```
rc = SQLConnect((HDBC)hdbc, (SQLCHAR*)dataSource, SQL_NTS,
(SQLWCHAR*)"user", SQL_NTS, (SQLWCHAR*)"password", SQL_NTS);
rc = SQLSetConnectAttr((HDBC)hdbc, SQL_ATTR_AUTOCOMMIT,
SQL_AUTOCOMMIT_OFF, SQL_NTS);
```

   where:

   user

   is the user name required to connect to the RM.

   password

   is the password required to connect to the RM.

2. Load the driver library and call the `GetXaSwitch()` function. `GetXaSwitch()` returns the XA switch data structure (`xa_switch_t`) exposed by the RM. `xa_switch_t` contains function pointers to the `xa_` routines that must be invoked by the TM to interact with the RM. For example:

```
#ifndef WIN32
        LPCSTR libFile = "ddora28.dll";
        HMODULE oraLib = LoadLibrary(libFile);
        xaSwitch = (XaSwitch)GetProcAddress(oraLib, "GetXaSwitch");
#else // Unix
        handle = dlopen("ddora28.so", RTLD_LAZY);
        xaSwitch = (XaSwitch)dlsym(handle, "GetXaSwitch_");
#endif // End of WIN32
        xaSwitch(0, &xaSwitchPtr);
```

3. Establish an XA connection between TM and RM. As a result, a global transaction is created. For example:

```
int RMID = rmid_value;
// Use xa_open_entry with XA connection string, RMID, and flag to establish the
connection.
rc = xaSwitchPtr->xa_open_entry("Oracle_XA+HostName=host_name+
PortNumber=1521+Sid=sid_value+ACC=account_credentials+
SesTM=session_timeout+LogDir=log_dir_path",rmid,flag);
```

   where:

   rmid_value

   is the RM identifier.

   host_name

   is the name or IP address of the RM you want to connect to.

   sid_value

   is the system identifier that refers to the instance of Oracle running on the server.

account_credentials

are the credentials required to connect to the RM. They must be provided in the following format: P*/USER/PASSWORD*.

session_timeout

is the number of seconds the session remains active.

log_dir_path

is the path to the directory where log files are stored.

flag

determines the function that is called after the connection ends. For example, TMASYNC and TMNOFLAGS.

4. Create an XID for the transaction branch you want to associate with the global transaction. For example:

```
void DoMakeXid(int id, XID *xaXid){
int *dataPtr;
memset(xaXid, 0, sizeof(*xaXid));
xaXid->formatID = format_id
xaXid->gtrid_length = gtrid_length;
xaXid->bqual_length = bqual_length;
dataPtr = (int*)&xaXid->data;
dataPtr[0] = GetCurrentThreadId();
dataPtr[1] = GetCurrentThreadId();
}
```

where:

format_id

is the identifier that indicates the naming convention used by the TM.

gtrid_length

is the length of the global transaction ID. It should not exceed 64 bytes.

bqual_length

is the length of the branch qualifier ID. It should not exceed 64 bytes.

Note: In the above example, the GetCurrentThreadId() function returns the global transaction and branch qualifier IDs. However, if you are using an Oracle TM, you can use the IDs obtained from the TM instead of those returned by the GetCurrentThreadId() function.

5. Start the transaction branch, enlist the connection, and then execute the required query.

```
// Use xa_start_entry to start the transaction branch.
rc = xaSwitchPtr->xa_start_entry(&xId, rmid, flag);

// Enlist connection.
rc = SQLSetConnectOption(hDbc, SQL_ATTR_ENLIST_IN_XA, (UDWORD)1);
```

```
// Execute query.
rc = SQLExecDirect(hstmt, (unsigned char*)"INSERT INTO table_name VALUES(value_1,
'value_2')", SQL_NTS);
```

6. Prepare and commit the transaction branch; then, delist the connection.

```
// Use xa_prepare_entry to prepare the trasaction branch.
rc = xaSwitchPtr->xa_prepare_entry(&xId, rmid, TMNOFLAGS);

// Use xa_commit_entry to commit the transaction branch.
rc = xaSwitchPtr->xa_commit_entry(&xId, rmid, TMNOFLAGS);

// Delist connection.
rc = SQLSetConnectOption(hDbc, SQL_ATTR_ENLIST_IN_XA, (UDWORD)0);
```

7. End the transaction branch.

```
// Use xa_end_entry to end the transaction branch.
rc = xaSwitchPtr->xa_end_entry(&xId, rmid, flag);
```

MTS support

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

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

OS authentication

Oracle has a feature called OS Authentication that allows you to connect to an Oracle database via the operating system user name and password. To connect, use a forward slash (/) for the user name and leave the password blank. To configure the Oracle server, refer to the Oracle server documentation. This feature is valid when connecting from a data source, a connection string, or a logon dialog box.

Isolation and lock levels supported

Oracle supports isolation level 1 (read committed) and isolation level 3 (serializable). Oracle supports record-level locking.

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

Unicode support

The Oracle Wire Protocol driver automatically determines whether the Oracle database is a Unicode database.

See also

Data types on page 24

Using parameter arrays

Beginning with Oracle 9i, Oracle databases natively support parameter arrays, and the Oracle Wire Protocol driver, in turn, supports them. When designing an application for performance, using native parameter arrays for bulk inserts or updates, for example, can improve performance.

Note: The Oracle Wire Protocol driver currently does not support the use of BLOB, CLOB, LONG, LONG RAW, and XMLType data types with array binding.

Refer to "Using arrays of parameters" in the *Progress DataDirect for ODBC Drivers Reference* for more information.

See also

Using bulk load for batch inserts on page 154

Support of materialized views

When connected to an Oracle 9i or higher server, the Oracle Wire Protocol driver supports the creation of materialized views. Materialized views are like any other database view with the following additions: the results are stored as a database object and the results can be updated on a schedule determined by the Create View statement.

Materialized views improve performance for data warehousing and replication. Refer to the Oracle documentation for more information about materialized views.

Stored procedure results

When you enable the Procedure Returns Results connection option, the driver is able to return result sets from stored procedures/functions. In addition, SQLGetInfo(SQL_MULT_RESULTS_SETS) returns Y and SQLGetInfo(SQL_BATCH_SUPPORT) returns SQL_BS_SELECT_PROC. If this option is enabled and you execute a stored procedure that does not return result sets, you incur a small performance penalty.

This feature permits stored procedures to return ref cursors. For example:

```
Create or replace package GEN_PACKAGE as
CURSOR G1 is select CHARCOL from GTABLE2;
type GTABLE2CHARCOL is ref cursor return G1%rowtype;
end GEN_PACKAGE;
```

```
Create or replace procedure GEN_PROCEDURE1 (
  rset IN OUT GEN_PACKAGE.GTABLE2CHARCOL, icol INTEGER) as
begin
   open rset for select CHARCOL from GTABLE2
     where INTEGERCOL <= icol order by INTEGERCOL;
end;
```

When executing the stored procedures with result sets, do not include the result set arguments (Oracle ref cursors) in the list of procedure parameters. The result set returned through the ref cursor is returned as a normal ODBC result set.

```
{call GEN_PROCEDURE1 (?)}
```

where ? is the parameter for the icol argument.

For more information, refer to your Oracle SQL documentation.

Note: When executing a stored procedure that returns both ref cursors and stored procedures, the driver returns ref cursors first, followed by implicit results.

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 (UNIX/Linux), the driver checks the IANAAppCodePage option. 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 Oracle server is running code page ISO-8859-P1.
- When you insert a Euro character (€) from the Windows client and then fetch it back, an upside down question mark (¿) is displayed on the client instead of the Euro symbol.

This substitution occurs because the Euro character does not exist within the characters defined by the ISO-8859-P1 character set on the Oracle server. The Oracle server records the code point for its substitution character in the table instead of the code point for the Euro. This code point is an upside down question mark in the Windows cp1252 code page.

This is not a driver error. The code page of the Oracle database could not recognize the Euro code point and used its substitution character in the table. The best way to avoid these problems is to use the same code page on both the client and server machines.

You can check the native code point stored in the Oracle database using SQL*Plus with a SQL statement similar to the following:

```
SELECT dump(columnname, 1016) FROM yourtable;
```

Check the returned hexadecimal values to verify whether the data you intended to reside in the table is there. If it appears that Oracle substituted a different code point, then check the Oracle database code page to see if your intended character exists. If your character does not exist in the code page, then no error is involved; Oracle simply does not recognize the original character, and uses its substitution character instead.

Using failover

To ensure continuous, uninterrupted access to data, the Progress DataDirect *for* ODBC driver provides 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
- Extended Connection Failover

- Select Connection Failover

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" for information about primary and alternate servers.

Note: The driver also supports configuring connection and extended connection failover using the `TNSNAMES.ORA` file. See "Configuring failover using the TNSNAMES.ORA file" for details.

See also

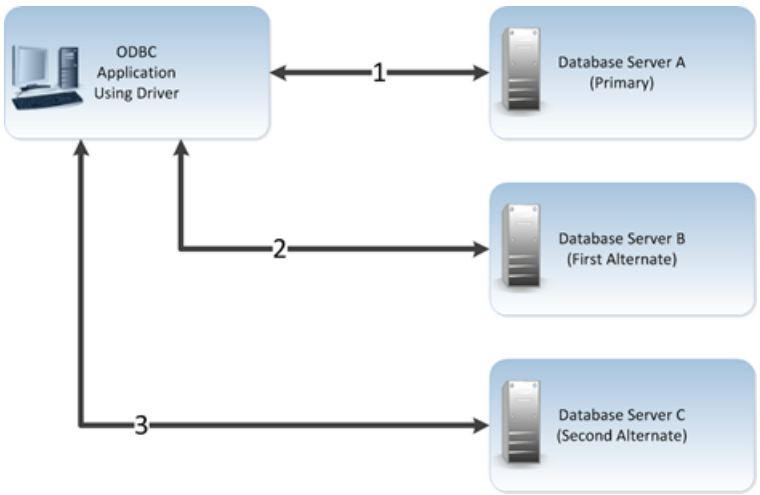
Failover Mode on page 210
Connection failover on page 108
Extended connection failover on page 109
Select connection failover on page 110
Alternate Servers on page 173
Guidelines for primary and alternate servers on page 111
Configuring failover using the TNSNAMES.ORA file on page 116

Connection failover

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 driver 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" 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 "Configuring Failover-Related Options."

See also

Using connection retry on page 112
Configuring failover-related options on page 113

Extended connection failover

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

- New connections, in the same way as described in "Connection Failover"
- 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.

See also

Connection failover on page 108
Failover Granularity on page 209
Failover Preconnect on page 210

Select connection failover

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

- New connections, in the same way as described in "Connection failover"
- Lost connections, in the same way as described in "Extended connection failover"

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.
```

See also

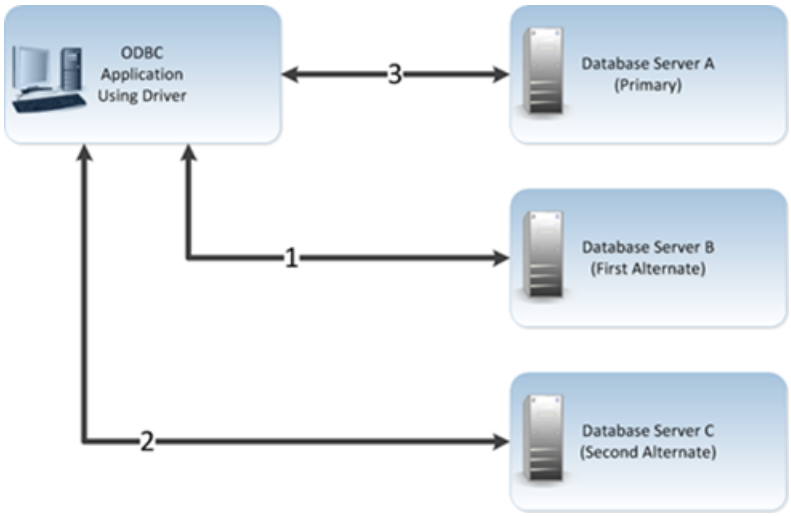
Connection failover on page 108
Extended connection failover on page 109

Guidelines for primary and alternate servers

Oracle databases provide advanced database replication technologies through the Oracle Real Application Clusters (RAC) feature. The failover functionality provided by the drivers does not require RAC, but can work with this technology to provide comprehensive failover protection. To ensure that failover works correctly, alternate servers should mirror data on the primary server or be part of a configuration where multiple database nodes share the same physical data.

Using client load balancing

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.

See also

Load Balancing on page 224

Using connection retry

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 "Configuring Failover-Related Options."

See also

Connection Retry Count on page 188
Connection Retry Delay on page 189
Configuring failover-related options on page 113

Configuring failover-related options

The following table summarizes how failover-related connection options work with the driver. See "Connection option descriptions" for details about configuring the options. The step numbers in the table refer the procedure that follows the table

Table 2: Summary: Failover and Related Connection Options

Option	Characteristic
Alternate Servers (AlternateServers) (See step 1 on page 114)	One or multiple alternate database servers. An IP address or server name identifying each server is required. **Default:** None
Connection Retry Count (ConnectionRetryCount) (See step 5 on page 115)	Number of times the driver retries the primary database server, and if specified, alternate servers until a successful connection is established. **Default:** `0`
Connection Retry Delay (ConnectionRetryDelay) (See step 6 on page 115)	Wait interval, in seconds, between connection retry attempts when the Connection Retry Count option is set to a positive integer. **Default:** 3
Failover Granularity (FailoverGranularity) (See step 3 on page 114)	Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection. 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)

Option	Characteristic
Failover Mode (FailoverMode) (See step 2 on page 114)	Specifies the type of failover method the driver uses. If set to `0` (Connection), the driver provides failover protection for new connections only. If set to `1` (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress. If set to `2` (Select), the driver provides failover protection for new and lost connections. In addition, it preserves the state of work performed by the last Select statement executed. **Default:** `0` (Connection)
Failover Preconnect (FailoverPreconnect) (See step 4 on page 115)	Specifies whether the driver tries to connect to the primary and an alternate server at the same time. 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)
Load Balancing (LoadBalancing) (See step 7 on page 115)	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. 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). **Default:** `0` (Disabled)

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.

See also

Connection option descriptions on page 163

A connection string example

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

```
DSN=AcctOracleServer;AlternateServers=(HostName=AccountingOracleServer:PortNumber=1521:
SID=Accounting, HostName=255.201.11.24:PortNumber=1522:ServiceName=ABackup.NA.MyCompany);
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 AcctOracleServer.

An odbc.ini file example

To configure the 32-bit 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/ivoraxx.so
Description=DataDirect Oracle Wire Protocol
...
AlternateServers=(HostName=MyServer:PortNumber=1521:SID=Accounting,
HostName=255.201.11.24:PortNumber=1522:ServiceName=ABackup.NA.MyCompany)
...
```

```
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.

Configuring failover using the TNSNAMES.ORA file

The driver supports the following types of failover for connections established using the `TNSNAMES.ORA` file:

- Connection failover
- Extended connection failover

For details, see "Connection failover" and "Extended connection failover".

To configure connection and extended connection failover using the `TNSNAMES.ORA` file, set the following parameters in the `TNSNAMES.ORA` file:

```
net_service_primary=
(DESCRIPTION=
(FAILOVER=on)
(ADDRESS=PROTOCOL=TCP)(HOST=hostname_primary)(PORT=portnumber_primary))
(ADDRESS=PROTOCOL=TCP)(HOST=hostname_alternate)(PORT=portnumber_alternate))
(CONNECT_DATA=
(FAILOVER_MODE=
(BACKUP=net_service_alternate)
(TYPE=session)
(RETRIES=number_of_retries)
(DELAY=delay_time)
(METHOD=basic))
(SERVICE_NAME = ASC)
)
)
```

where:

net_service_primary

specifies the name of the net service that contains the parameters required for configuring failover.

hostname_primary

specifies the name or IP address of the primary server to which the driver attempts to connect.

portnumber_primary

specifies the port number of the primary server listener.

hostname_alternate

specifies the name or IP address of the alternate server. If the attempt to connect to the primary server fails, the driver attempts to connect to the alternate server.

portnumber_alternate

specifies the port number of the alternate server listener.

net_service_alternate

specifies the name of the net service that provides the alternate server details for supporting extended connection failover.

number_of_retries

specifies the number of times the driver attempts to connect to the specified servers after the initial unsuccessful connection attempt. In each attempt, the driver first tries to connect to the primary server and then to the alternate server.

delay_time

specifies the amount of time in seconds that the driver has to wait for before reattempting to connect to the specified servers. In each attempt, the driver first tries to connect to the primary server and then to the alternate server.

The following example shows how to configure connection and extended connection failover using the TNSNAMES.ORA file.

```
NetService1=
(DESCRIPTION =
(FAILOVER=on)
(ADDRESS=PROTOCOL=TCP)(HOST=server3)(PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=server4)(PORT=1521))
(CONNECT_DATA=
(FAILOVER_MODE=
(BACKUP=NetService2)
(TYPE=session)
(RETRIES=2)
(DELAY=5)
(METHOD=basic))
(SERVICE_NAME=ASC)
)
)
```

See also

Connection failover on page 108
Extended connection failover on page 109

Using client information

Many databases allow applications to store client information associated with a connection. For example, the following types of information can be useful for database administration and monitoring purposes:

- Name of the application currently using the connection.

- User ID for whom the application using the connection is performing work. The user ID may be different than the user ID that was used to establish the connection.
- Host name of the client on which the application using the connection is running.
- Product name and version of the driver on the client.
- Additional information that may be used for accounting or troubleshooting purposes, such as an accounting ID.

For Oracle 11*g* R2 and higher, this information is managed through the client information feature.

See "How databases store client information" for more information about how Oracle stores client information.

See also

How databases store client information on page 118

How databases store client information

Typically, databases that support storing client information do so by providing a register, a variable, or a column in a system table in which the information is stored. If an application attempts to store information and the database does not provide a mechanism for storing that information, the driver caches the information locally. Similarly, if an application returns client information and the database does not provide a mechanism for storing that information, the driver returns the locally cached value.

Storing client information

Your application can store client information associated with a connection. The following table shows the driver connection options that your application can use to store client information and where that client information is stored for each database. See "Connection Option Descriptions" for a description of each option.

Table 3: Database Locations for Storing Client Information

Option	Description	Location
Accounting Info (AccountingInfo)	Additional information that may be used for accounting or troubleshooting purposes, such as an accounting ID	CLIENT_INFO value in the V$SESSION table.
Action (Action)	The current action within the current module.	ACTION value in the V$SESSION table.
Application Name (ApplicationName)	Name of the application currently using the connection	CLIENT_IDENTIFIER attribute. In addition, this value is also stored in the PROGRAM value in the V$SESSION table.
Client Host Name (ClientHostName)	Host name of the client on which the application using the connection is running	MACHINE value in the V$SESSION table.
Client ID (ClientID)	Additional information about the client	CLIENT_IDENTIFIER value in the V$SESSION table.
Client User (ClientUser)	User ID for whom the application using the connection is performing work	OSUSER value in the V$SESSION table.

Option	Description	Location
Module (Module)	The name of a stored procedure or the name of the application	MODULE value in the V$SESSION table.
Program ID (ProgramID)	Product name and version of the driver on the client	PROCESS value in the V$SESSION table.

See also

Connection option descriptions on page 163

Using security

The driver supports 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 driver supports 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. The Oracle Wire Protocol driver 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 requirements

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

Table 4: Kerberos Authentication Requirements for the Oracle 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 one of the following databases: • Oracle 12*c* (R1 and R2) • Oracle 11*g* (R1 and R2) • Oracle 10*g* (R1 and R2) • Oracle 9*i* (R2) In addition, Oracle Advanced Security 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 on one of the following operating systems: Windows Server 2003 or Windows 2000 Server Service Pack 3 or higher • MIT Kerberos 1.4.2 or higher
Client	The client must be administered by the same domain controller that administers the database server.

Kerberos authentication

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

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

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

If the application uses Kerberos authentication from a UNIX and 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.

 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.

OS authentication

On all supported platforms, Oracle has a feature called OS Authentication that allows you to connect to an Oracle database via the operating system user name and password. To connect, use a forward slash (/) for the user name and leave the password blank. To configure the Oracle server, refer to the Oracle server documentation. This feature is valid when connecting from a data source, a connection string, or a logon dialog box.

Oracle Internet Directory (OID)

Oracle Internet Directory (OID) is an LDAP-based directory service that is used for the storage, retrieval and administration of collections of object information. Oracle Internet Directory is often used as a single sign-on solution because of its ability to centrally store authentication information and permissions for Oracle databases. The driver supports authenticating with Oracle Internet Directory without additional driver configuration.

Oracle Wallet SSL Authentication

The driver supports Oracle Wallet SSL authentication, which was introduced in Oracle 11.1.0.6. When Oracle Wallet SSL Authentication is enabled, SSL certificates are authenticated against a list of trusted certificates stored in the wallet. Refer to the documentation for your Oracle database for detailed information on the Oracle Wallet feature.

To enable Oracle Wallet SSL authentication:

- Enable SSL (`EncryptionMethod=1`).
- Set the Authentication Method connection option:
  - If a user ID or password is **not** required, set to `11` (SSL).
  - If a user ID or password is required, set to `12` (SSL with UID & PWD).
- Set the Key Store option to specify the absolute path of the keystore file in your wallet that contains the SSL certificate information.
- Optionally, if you are using a file in the PKCS#12 format, set the Key Store Password option to specify the password if required by your environment.
- Set the Trust Store option to specify the absolute path of the truststore file in your wallet that contains the SSL certificate information.
- Optionally, if you are using a file in the PKCS#12 format, set the Trust Store Password option to specify the password if required by your environment.
- If a user ID and password is required (`AuthenticationMethod=12`), specify the corresponding value for the User Name and Password options.

Note: When Oracle Wallet SSO is used as the Key Store or Trust Store, the Key Store Password and Trust Store Password options are not required.

See also

Authentication Method on page 176
Encryption Method on page 206
Key Store on page 218

Key Store Password on page 219
Trust Store on page 254
Trust Store Password on page 255
User Name on page 256
Password on page 232

Oracle Wallet Password Store

Oracle Wallet Password Stores allow the driver to retrieve database credentials from an Oracle Wallet to be used when authenticating to the server. Using Oracle Wallet Password Stores simplifies password management by centrally storing database credential information, thereby providing a method to modify the user ID and password without changing application code. In addition, by storing credentials in a wallet, security is improved by eliminating the need include passwords in the application code or scripts.

When this feature is enabled, the driver retrieves the user ID and password for a database from the Oracle Wallet file specified by the Credentials Wallet Path (CredentialsWalletPath) option. Since multiple sets of database credentials can be stored in a wallet file, the driver retrieves only the user name and password associated with the string specified by the CredentialsWalletEntry (CredentialsWalletEntry). After the user ID and password are retrieved, the driver uses these credentials to authenticate to the server.

Entries for data base connection credentials in a wallet are created using the following syntax from a command line:

```
mkstore -wrl <credentials_wallet_path> -createCredential <credentials_wallet_entry>
 <userID> <password>
```

From these entries, you can determine the values for the Credentials Wallet Path and Credentials Wallet Entry options when configuring the driver.

To enable authentication using a Oracle Wallet password store:

- Set the Authentication Method (AuthenticationMethod) option to `16` (Wallet UID & PWD).
- Set the Credentials Wallet Path (CredentialsWalletPath) option to specify the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. The driver supports `ewallet.p12` and `cwallet.sso` files for wallets.
- Set the Credentials Wallet Entry (CredentialsWalletEntry) to specify the string value used to identify database credential information stored in your Oracle Wallet. This value is defined when creating or modifying credentials stored in a wallet and is typically a net service name, Oracle service name, or *`host`*:*`port`*:*`SID`* string, but can be any value specified by the user. Credentials Wallet Entry provides a method to retrieve the correct credentials when multiple user name and password pairs are stored in a wallet file.
- If you are using an `ewallet.p12` file for your wallet, set the Wallet Password (CredentialsWalletPassword) to specify the password used to access the Oracle Wallet in which your database credential information is stored. The wallet password is typically configured when the wallet is created.

  Note: On the GUI, the Wallet Password is exposed on the Logon dialog.

Note:

- When using an Oracle Wallet password store (`AuthenticationMethod=16`), specifying values for the User Name (LogonID) or Password (Password) options returns a warning and the values are ignored.
- If you are using an `cwallet.sso` file, you do no need to specify a value for the Wallet Password option. The password for the wallet is stored in this file and, therefore, no value for this option needs to be provided.

See also

Authentication Method on page 176
Credentials Wallet Path on page 191
Credentials Wallet Entry on page 190
Wallet Password on page 257

Entra ID access token authentication

The driver supports authenticating to the server using an access token obtained from Microsoft Entra ID (Azure Active Directory). The driver encrypts data using TLS/SSL encryption when using Entra ID access token authentication; therefore, the Encryption Method (EncryptionMethod) option is automatically set to `1` (SSL) when Entra ID authentication is enabled.

To enable authentication Entra ID access token authentication:

- Set the Host (Hostname) option to specify either the IP address in IPv4 or IPv6 format or the server name for your Oracle server.
- Set the Port Number (PortNumber) option to specify the TCP port of the primary database server that is listening for connections to the database.
- Set the Authentication Method (AuthenticationMethod) option to `38` (EntraIDAccessToken).
- Set the Entra Access Token (EntraAccessToken) option to specify the access token that you have obtained from Entra ID.

  Note: If you are using the Setup dialog, the Entra Access Token is exposed on the logon dialog box when Authentication Method is set to **EntraIDAccessToken**.

- Set the Encryption Method (EncryptionMethod) to `1` (SSL).

The following examples show the connection information required to establish a session with Entra ID access token authentication.

Connection string

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;AuthenticationMethod=38;
       EncryptionMethod=1;EntraAccessToken="access_token";
       HostName=myserver;PortNumber=1521;
```

odbc.ini

```
[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora28.so
...
Description=DataDirect 8.0 Oracle Wire Protocol
...
AuthenticationMethod=38
...
EncryptionMethod=1
...
EntraAccessToken="access_token"
...
HostName=myserver
...
PortNumber=1521
...
```

See also

Authentication Method on page 176
Host on page 213
Port Number on page 233
Entra Access Token on page 207
Encryption Method on page 206

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.

Your Progress DataDirect *for* ODBC driver supports Transport Layer Security (TLS) and Secure Sockets Layer (SSL). TLS/SSL are industry-standard protocols for sending encrypted data over database connections. TLS/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.

Data encryption and integrity

The driver supports the following types of data encryption:

- TLS/SSL
- Oracle Advanced Security

In addition, the Oracle driver supports Oracle Advanced Security data integrity checks.

TLS/SSL encryption

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

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

The handshake involves the following types of authentication:

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

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

Certificates

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

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

TLS/SSL server authentication

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

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

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

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

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

- Set the Host Name (HostName) option to specify the name or the IP address of the server to which you want to connect.
- Set the Port Number (PortNumber) option to specify the port number of the server listener. The default is `1522`.
- Set the Service Name (ServiceName) option to specify the Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Trust Store (Truststore) option to specify either the full path of the truststore file or the contents of the TLS/SSL certificates.

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

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

Note:

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

The following examples show how to configure the driver to establish a connection via user ID/password authentication and use data encryption via TLS/SSL server authentication. In these examples, 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.

Connection string

Truststore:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;EncryptionMethod=1;
EnableFIPS=1;HostName=YourServer;HostNameInCertificate=MySubjectAltName;
PortNumber=1522;ServiceName=SALES.US.ACME.COM;Truststore=TrustStoreName;
ValidateServerCertificate=1;
```

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

Windows certificate store:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;EncryptionMethod=1;
EnableFIPS=1;HostName=YourServer;HostNameInCertificate=MySubjectAltName;
PortNumber=1522;ServiceName=SALES.US.ACME.COM;ValidateServerCertificate=1;
```

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

odbc.ini

Truststore:

```
Driver=ODBCHOME/lib/ivoraXX.so
Description=DataDirect Oracle Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=1522
...
ServiceName=SALES.US.ACME.COM
...
Truststore=TrustStoreName
...
ValidateServerCertificate=1
...
```

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

Windows certificate store:

```
Driver=ODBCHOME/lib/ivoraXX.so
Description=DataDirect Oracle Wire Protocol
...
EnableFIPS=1
...
EncryptionMethod=1
...
HostName=YourServer
...
HostNameInCertificate=MySubjectAltName
...
PortNumber=1522
...
ServiceName=SALES.US.ACME.COM
...
ValidateServerCertificate=1
...
```

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

See also

Importing root certificates into the Windows certificate store on page 129
Trust Store on page 254
Using SQL_COPT_INMEMORY_TRUSTSTORECERT on page 128
Connection option descriptions on page 163
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 132

Using SQL_COPT_INMEMORY_TRUSTSTORECERT

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

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

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

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

...

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

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

See also

Trust Store on page 254

Importing root certificates into the Windows certificate store

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

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

Importing root certificates using Certificate Import Wizard

To import root certificates using Certificate Import Wizard:

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

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

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

Importing root certificates using a PowerShell script

To import root certificates using a PowerShell script:

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

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

where:

`truststore_password`

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

`truststore_filepath`

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

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

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

TLS/SSL client authentication

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

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

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

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

- Set the Host Name (HostName) option to specify the name or the IP address of the server to which you want to connect.
- Set the Port Number (PortNumber) option to specify the port number of the server listener. The default is `1522`.
- Set the Service Name (ServiceName) option to specify the Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name.
- Set the Encryption Method (EncryptionMethod) option to `1`.
- Set the Validate Server Certificate (ValidateServerCertificate) option to determine whether the driver validates the certificates sent by the server. When it is set to `1`, the driver validates the certificates. When it is set to `0`, the driver does not validate the certificates.
- Set the Host Name In Certificate (HostNameInCertificate) option to specify the host name that is specified in the Subject of the certificate. This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested. Consult your SSL administrator for the correct value.
- Set the Key Store (Keystore) option to specify the location of the keystore file.
- Set the Keystore Password (KeystorePassword) option to specify the password that is used to access the keystore file.
- Optionally, set the Enable FIPS (EnableFIPS) connection option to `1` to allow the driver to load the FIPS provider. The FIPS provider contains a set of approved cryptographic algorithms that conform to the Federal Information Processing Standards (FIPS) specified in FIPS 140-2. If you do not specify a value for Enable FIPS, the driver uses its default value (`0`) and loads the default provider.

Note:

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

The following examples show how to configure the driver to establish a connection via user ID/password authentication and use data encryption via TLS/SSL client authentication. In these examples, since `ValidateServerCertificate=1` and `EnableFIPS=1`, the driver validates the certificate sent by the server and the host name specified by HostNameInCertificate, and loads the FIPS provider for data encryption.

Connection string

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;EnableFIPS=1;
EncryptionMethod=1;HostName=YourServer;HostNameInCertificate=MySubjectAltName;
PortNumber=1522;ServiceName=SALES.US.ACME.COM;Keystore=KeyStoreName;
ValidateServerCertificate=1;
```

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

odbc.ini

```
Driver=ODBCHOME/lib/ivoraXX.so
Description=DataDirect Oracle Wire Protocol
...
EnableFIPS=1;
...
EncryptionMethod=1;
...
HostName=YourServer;
...
HostNameInCertificate=MySubjectAltName;
...
PortNumber=1522;
...
ServiceName=SALES.US.ACME.COM;
...
KeyStore=KeyStoreName;
...
ValidateServerCertificate=1;
...
```

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

See also

Connection option descriptions on page 163
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 132

Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms

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

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

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

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

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

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

Truststore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`truststore_filename`

is the name of the truststore file.

`cryptographic_algorithm`

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

`truststore_password`

is the password required for accessing the truststore file.

Example:

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

Keystore:

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

where:

`certificate_name`

is the name of the certificate you are generating.

`privatekey_file`

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

`truststore_filename`

is the name of the keystore file.

`cryptographic_algorithm`

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

Example:

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

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

Designating an OpenSSL library

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

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

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

File replacement

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

Windows:

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

UNIX/Linux:

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

Designating the absolute path to a library

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

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

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

See also

CryptoLibName on page 192
SSLLibName on page 249

Using Oracle Wallet as a keystore

The driver supports the use of Oracle Wallet as a keystore and truststore. A wallet is a password-protected container that is created using the Oracle Wallet Manager. It contains trusted certificates for authenticating the server's public certificate. The wallet may also contain client private key and associated certificates required for client authentication.

Depending on the contents of your Oracle Wallet, you must provide values for specific connection options as described in the following scenarios:

- If a wallet contains client certificates, you must specify a value for the Key Store connection option. If you are using a file in the PKCS#12 format, you must also specify a value for the Key Store Password option.
- If a wallet contains the trusted certificates and client certificates required for both server and client authentication, you must specify values for only the Trust Store connection option. If you are specifying a file in the PKCS#12 format, you must also specify a value for the Trust Store Password option. The driver treats the truststore file as a keystore and loads client certificates required for client authentication.

Oracle Wallet is compliant with PKCS#12 and SSO formats.

See also

Key Store on page 218
Key Store Password on page 219
Trust Store on page 254
Trust Store Password on page 255

Oracle Advanced Security

To enable support for TLS/SSL connections to Oracle, the Oracle database must be configured with the Oracle Advanced Security bundle. This is an option available from Oracle as an add-on to Oracle Enterprise Edition Servers.

The driver also supports encryption and data integrity checks through Oracle Advanced Security. Oracle Advanced Security provides the Advanced Encryption Standard (AES), DES, 3DES, and RC4 symmetric cryptosystems for protecting the confidentiality of network traffic.

Encrypting network data provides data privacy so that unauthorized parties cannot view and alter clear text data as it passes over the network. Attacks on intercepted data include data modification and replay attacks.

- In a data modification attack, an unauthorized party intercepts transmitted data, alters it, and retransmits it. For example, suppose a customer order for 5 widgets for delivery to an office in San Francisco is intercepted. A data modification attack might change the quantity to 500 and the delivery address to a warehouse in Los Angeles, and then retransmit the order.
- In a replay attack, a set of valid data is retransmitted a number of times. For example, an order for 100 widgets is intercepted and then retransmitted ten times so the final order quantity equals 1,000 widgets.

Because data integrity protection operates independently from the encryption process, you can enable data integrity with or without enabling encryption.

Summary of security-related options

The following table summarizes how security-related connection options work with the drivers. The connection options are listed alphabetically by the GUI name that appears on the driver Setup dialog box. The connection string attribute name is listed immediately after the GUI name in parentheses. See "Connection option descriptions" for details about configuring the options.

Table 5: Summary: Authentication Connection Options

Option	Description
Authentication Method (AuthenticationMethod)	Specifies the method the driver uses to authenticate the user to the server when a connection is established. If set to 1 (Encrypt Password), the driver sends the user ID in clear text and an encrypted password to the server for authentication. If set to 3 (Client Authentication), the driver uses client authentication when establishing a connection. The database server relies on the client to authenticate the user and does not provide additional authentication. If set to 4 (Kerberos Authentication), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments. When set to 5 (Kerberos with UID & PWD), the driver uses both Kerberos authentication and user ID and password authentication. The driver first authenticates the user using Kerberos. If a user ID and password are specified, the driver reauthenticates using the user name and password supplied. An error is generated if a user ID and password are not specified. If set to 6 (NTLM), the driver uses NTLMv1 authentication for Windows clients. If set to 11 (SSL), the driver uses SSL certificate information to authenticate the client with the server when using Oracle Wallet. The User Name and Password options should **not** be specified. See "Oracle Wallet SSL Authentication" for additional requirements. If set to 12 (SSL with UID & Password), the driver uses user ID/password and SSL authentication to connect with the server when using Oracle Wallet. See "Oracle Wallet SSL Authentication" for additional requirements. If set to 16 (Wallet UID & PWD), the driver authenticates to the server using a user ID and password retrieved from Oracle Wallet. See "Oracle Wallet Password Store" for additional requirements. If set to 38 (EntraIDAccessToken), the driver authenticates to the server using an Entra ID access token. This setting requires the Entra Access Token option to be specified. If an access token is not specified, the driver throws an exception. All communications with the service are encrypted using TLS/SSL encryption. **Default:** 1 (Encrypt Password)

Option	Description
Credentials Wallet Entry (CredentialsWalletEntry)	Specifies the string value used to identify database credential information stored in an Oracle Wallet. When Authentication Method is set to 16 (Wallet UID & PWD), the driver retrieves the user ID and password associated with the specified value from the wallet and uses them to authenticate to the server. This value provides a method for the correct user ID and password to be retrieved when there are multiple pairs in a wallet. See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.
Wallet Password (CredentialsWalletPassword)	Specifies the password used to access the Oracle Wallet in which your database credential information is stored. When Authentication Method is set to 16 (Wallet UID & PWD), the driver uses this value to retrieve the database user ID and password that is stored in the wallet file specified by the Credentials Wallet Path option. See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.
Credentials Wallet Path (CredentialsWalletPath)	Specifies the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. When Authentication Method is set to 16 (Wallet UID & PWD), the driver retrieves the database user name and password from this file. See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.
Entra Access Token (EntraAccessToken)	Specifies the access token required to authenticate to an Oracle instance when using Entra ID access token authentication (`AuthenticationMethod=38`). Refer to the Oracle documentation for more information on obtaining an access token.
GSS Client Library (GSSClient)	The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC). **Default:** `native` (The driver uses the GSS client shipped with the operating system.)

Option	Description
ImpersonateUser (ImpersonateUser)	Specifies the proxy user ID used for impersonation. The value for Impersonate User determines your identity and permissions when executing queries. When a value is specified for this option, the driver authenticates according to the setting of the Authentication Method option; then, after establishing a connection, the driver attempts to reauthenticate as the destination user. Note that the administrator must grant CONNECT THROUGH permission to the authenticated user in order to impersonate the destination user; otherwise, an error is returned. **Default:** None
User Name (LogonID)	The default user ID that is used to connect to your database. **Default:** None

Table 6: Summary: Data Encryption Connection Options

Option	Description
Crypto Protocol Version (CryptoProtocolVersion)	Specifies the cryptographic protocols to use when TLS/SSL is enabled using the Encryption Method connection option (`EncryptionMethod=1`). **Default:** `TLSv1.3,TLSv1.2`
CryptoLibName (CryptoLibName)	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. **Default:** Empty string
DataIntegrityLevel (DIL)	Specifies a preference for the data integrity to be used on data sent between the driver and the database server. The connection fails if the database server does not have a compatible integrity algorithm. If set to `0` (Rejected), a data integrity check on data sent between the driver and the database server is refused. The connection fails if the database server specifies REQUIRED. If set to `1` (Accepted), a data integrity check can be made on data sent between the driver and the database server. Data integrity is used if the database server requests or requires it. If set to `2` (Requested), the driver enables a data integrity check on data sent between the driver and the database server if the database server permits it. If set to `3` (Required), a data integrity check must be performed on data sent between the driver and the database server. The connection fails if the database server specifies REJECTED. See "Encryption and Data Integrity" for more information. **Default:** `1` (Accepted)

Option	Description
DataIntegrityTypes (DIT)	Determines the method the driver uses to protect against attacks that intercept and modify data being transmitted between the client and server. You can enable data integrity protection without enabling encryption. If multiple values are specified and Oracle Advanced Security data integrity is enabled using the Data Integrity Level option, the database server determines which algorithm is used based on how it is configured. **Default:** `MD5, SHA1, SHA256, SHA384, SHA512`
Enable FIPS on page 200	Determines whether the OpenSSL library uses cryptographic algorithms from the FIPS provider or the default provider when TLS/SSL encryption is enabled (`Encryption Method=1`). If disabled, the OpenSSL library uses cryptographic algorithms from the default provider. If enabled, the OpenSSL library uses cryptographic algorithms from the FIPS provider. **Default:** Disabled
EncryptionLevel (EL)	Specifies a preference on whether to use encryption on data being sent between the driver and the database server. If set to `0` (Rejected), or if no match is found between the driver and server encryption types, data sent between the driver and the database server is not encrypted or decrypted. The connection fails if the database server specifies REQUIRED. If set to `1` (Accepted), encryption is used on data sent between the driver and the database server if the database server requests or requires it. If set to `2` (Requested), data sent between the driver and the database server is encrypted and decrypted if the database server permits it. If set to `3` (Required), data sent between the driver and the database server must be encrypted and decrypted. The connection fails if the database server specifies REJECTED. **Default:** `1` (Accepted)
Encryption Method (EncryptionMethod)	The method the driver uses to encrypt data sent between the driver and the database server. If set to `0` (No Encryption), data is not encrypted. If set to `1` (SSL), data is encrypted using the TLS/SSL protocols specified in the Crypto Protocol Version connection option. **Default:** `0` (No Encryption)
EncryptionTypes (ET)	Specifies the encryption algorithms to use if Oracle Advanced Security encryption is enabled using the Encryption Level connection property. **Default:** All listed encryption algorithms are selected.

Option	Description
Host Name In Certificate (HostNameInCertificate)	A host name for certificate validation when TLS/SSL encryption is enabled (`Encryption Method=1`) and validation is enabled (`Validate Server Certificate=1`). **Default:** None
Key Password (KeyPassword)	Specifies the password used to access the individual keys in the keystore file when TLS/SSL is enabled (`Encryption Method=1`) and TLS/SSL client authentication is enabled on the database server. **Default:** None
Key Store (Keystore)	The absolute path of the keystore file to be used when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/SSL client authentication is enabled on the database server. **Default:** None
Key Store Password (KeystorePassword)	The password used to access the keystore file when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/SSL client authentication is enabled on the database server. **Default:** None
SSLLibName (SSLLibName)	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 library contains the implementations of TLS/SSL protocols the driver uses for data encryption. **Default:** Empty string
Trust Store (Truststore)	The absolute path of the truststore file to be used when TLS/SSL is enabled (`EncryptionMethod=1`) and server authentication is used. **Default:** None

Option	Description
Trust Store Password (TruststorePassword)	Specifies the password that is used to access the truststore file when TLS/SSL is enabled (`EncryptionMethod=1`) and server authentication is used. **Default:** None
Validate Server Certificate (ValidateServerCertificate)	If 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 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:** Enabled

See also

Connection option descriptions on page 163
Oracle Wallet SSL Authentication on page 121
Oracle Wallet Password Store on page 122

Using DataDirect Connection Pooling

Supported on Windows, UNIX, and Linux only.

The Oracle Wire Protocol driver supports DataDirect Connection Pooling on Windows, UNIX, and Linux platforms. 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 driver enables 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 "Summary of pooling-related options" for details about how the connection options manage DataDirect connection pooling.

Important: On Windows, do not use connection pooling for the Windows Driver Manager at the same time as DataDirect connection pooling.

See also

Summary of pooling-related options on page 144

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 7: 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.

Your Progress DataDirect for ODBC driver handles 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" 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.

See also

Connection failover on page 108

Connection pool statistics

Progress DataDirect has created a connection attribute to monitor the status of the Progress DataDirect 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 8: Pool Statistics Connection Attribute

Connection Attribute	Description
SQL_ATTR_POOL_INFO Value: SQL_GET_POOL_INFO	Calling SQLGetConnectAttr (SQL_ATTR_POOL_INF, 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" for additional details about configuring the options.

Table 9: Summary: Connection Pooling Connection Options

Option	Characteristic
Connection Pooling (Pooling)	Specifies whether to use the driver's connection pooling. If set to `1` (Enabled), the driver uses connection pooling. If set to `0` (Disabled), the driver does not use connection pooling. **Default:** `0` (Disabled)
Connection Reset (ConnectionReset)	Determines whether the state of connections that are removed from the connection pool for reuse by the application is reset to the initial configuration of the connection.If set to `1` (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 `0` (Disabled), the state of connections is not reset. **Default:** `0` (Disabled)
Load Balance Timeout (LoadBalanceTimeout)	An integer value to specify the amount of time, in seconds, to keep connections open in a connection pool. Default: `0`

Option	Characteristic
Max Pool Size (MaxPoolSize)	An integer value to specify the maximum number of connections within a single pool. **Default:** `100`
Min Pool Size (MinPoolSize)	An integer value to specify the minimum number of connections that are opened and placed in a connection pool when it is created. If set to `0`, no connections are opened in addition to the current existing connection. **Default:** `0`

See also

Connection option descriptions on page 163

Using DataDirect Bulk Load

Supported on Windows, UNIX, and Linux only.

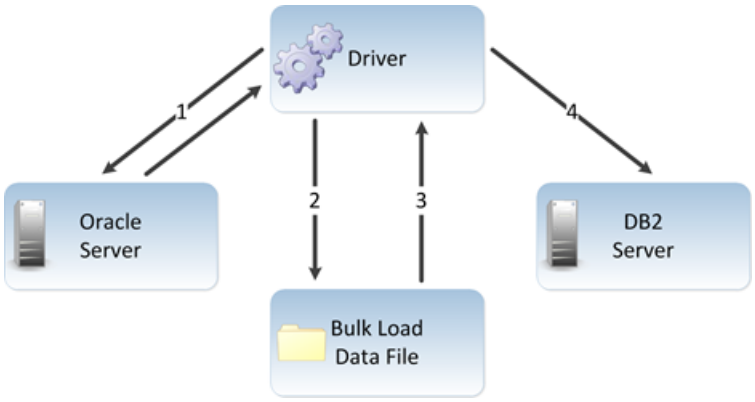
On Windows, UNIX, and Linux, the driver supports DataDirect Bulk Load when connected to Oracle databases version 9*i* R2 and higher. This feature allows your application to send large numbers of rows of data to a database. The driver sends the data to the database in a continuous stream instead of numerous 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 drivers. If the drivers are installed with an evaluation license, the bulk load feature is available for prototyping with your applications, but with limited scope. Contact your sales representative or Progress DataDirect SupportLink for further information.

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

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

Suppose that you had customer data on an Oracle server and need to export it to a DB2 server. The driver would perform the following steps:



1. Application using Oracle Wire Protocol driver sends query to and receives results from Oracle 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 DB2 server.

Bulk export and load methods

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

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

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

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

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

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

This is detailed in the code samples that follow.

See also

Bulk tab on page 73

Exporting data from a database

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

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

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

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

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

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

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

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

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

rc = SQLGetInfo (hdbc, SQL_DRIVER_HLIB, &hmod, 0, NULL);
if (rc != SQL_SUCCESS) {
    ODBC_error (henv, hdbc, SQL_NULL_HSTMT);
    EnvClose (henv, hdbc);
    exit (255);
}
    exportTableToFile = (PExportTableToFile)
    resolveName (hmod, "ExportTableToFile");
if (! exportTableToFile) {
    printf ("Cannot find ExportTableToFile!\n");
    exit (255);
}

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

```
        driverError (driverHandle, hmod);
}
```

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

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

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

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

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

See also

The bulk load configuration file on page 149
External overflow files on page 152

Bulk loading to a database

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

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

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

/* Get the driver's connection handle from the DM.
```

```
   This handle must be used when calling directly into the driver.*/

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

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

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

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

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.

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

See also

Bulk tab on page 73

The bulk load configuration file

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

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

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

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

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

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

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

See also

Exporting data from a database on page 147

Bulk load configuration file schema for Oracle

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

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

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

Verification of the bulk load configuration file

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

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

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

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

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

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

See also

Bulk tab on page 73

Sample applications

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

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

Character set conversions

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

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

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

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

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

See also

External overflow files on page 152

External overflow files

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

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

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

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

See also

Character set conversions on page 152

Limitations

- A bulk operation is not allowed in a manual transaction if it is not the first event.
- Once a bulk operation is started, any non-bulk operation is disallowed until the transaction is committed.
- Because of Oracle limitations, issuing a SELECT statement to determine a row count may return different results before and after a bulk load operation.

Summary of related options for DataDirect Bulk Load

Connection Options: Bulk	Description
Batch Size (BulkLoadBatchSize)	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. Default: `1024`
Bulk Binary Threshold (BulkBinaryThreshold)	The maximum size, in KB, of binary data that is exported to the bulk data file. If set to `-1`, all binary data, regardless of size, is written to the bulk data file, not to an external file. If set to `0`, all binary data, regardless of size, is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file. If set to *`x`*, any binary data exceeding this specified number of KB is written to an external file, not the bulk data file. A reference to the external file is written to the bulk data file. **Default:** None

Connection Options: Bulk	Description
Bulk Character Threshold (BulkCharacterThreshold)	The maximum size, in KB, of character data that is exported to the bulk data file. 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`
Bulk Load Options (BulkLoadOptions)	Toggles options for the bulk load process. If set to `128` (no index errors is enabled), the driver stops a bulk load operation when a value that would cause an index to be invalidated is loaded. For example, if a value is loaded that violates a unique or non-null constraint, the driver stops the bulk load operation and discards all data being loaded, including any data that was loaded prior to the problem value. If disabled, the bulk load operation continues even if a value that would cause an index to be invalidated is loaded. **Default:** Disabled
Field Delimiter (BulkLoadFieldDelimiter)	Specifies the character that the driver will use to delimit the field entries in a bulk load data file. **Default:** `,` (comma)
Record Delimiter (BulkLoadRecordDelimiter)	Specifies the character that the driver will use to delimit the record entries in a bulk load data file. **Default:** None

See "Connection option descriptions" for details about configuring the options.

See also

Connection option descriptions on page 163

Using bulk load for batch inserts

The driver uses the native bulk load protocol for database connections when the Enable Bulk Load connection option is set to `true` (enabled). 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.

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 a warning or an error. In this situation, the driver falls back to using standard parameter arrays.

Limitations

The driver supports native parameter arrays in Oracle 9i and higher databases with the following limitations:

- A bulk operation is not allowed in a manual transaction if it is not the first event.
- Bulk inserts into views are not allowed.
- Once a bulk operation is started, any non-bulk operation is disallowed until the transaction is committed.
- The Oracle Wire Protocol driver currently does not support the use of BLOB, CLOB, LONG, LONG RAW, and XMLType data types when using bulk load for parameter array batch.
- Because of Oracle limitations, issuing a SELECT statement to determine a row count may return different results before and after a bulk load operation.
- Oracle does not support literal values in a bulk load operation. You must use parameter markers for all columns being loaded.
- INSERT INTO SELECT statements are not supported.

Summary of related options for bulk load for batch inserts

Connection Options: Bulk	Description
Batch Size (BulkLoadBatchSize)	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. Default: `1024`
Enable Bulk Load (EnableBulkLoad)	Specifies the bulk load method. If 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 disabled, the driver uses standard parameter arrays. **Default:** Disabled

See "Connection option descriptions" for details about configuring the options.

See also

Connection option descriptions on page 163

Persisting a result set as an XML data file

DataDirect 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
```

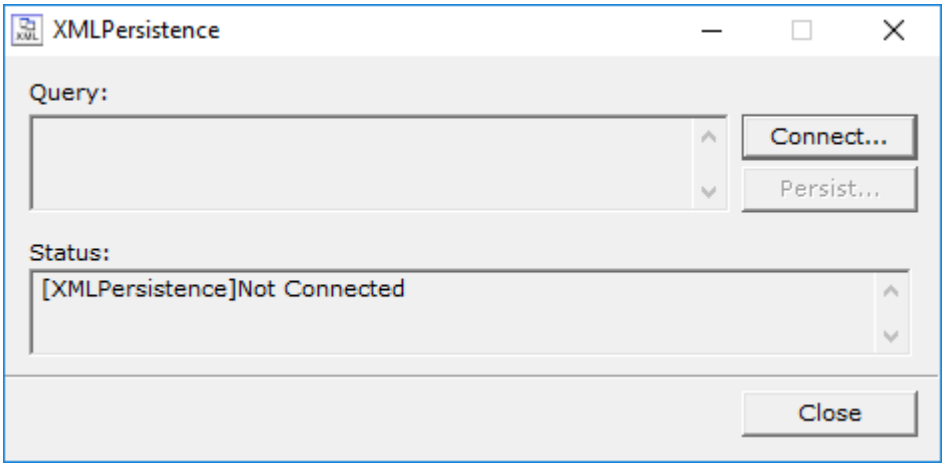
Using the Windows XML persistence demo tool

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

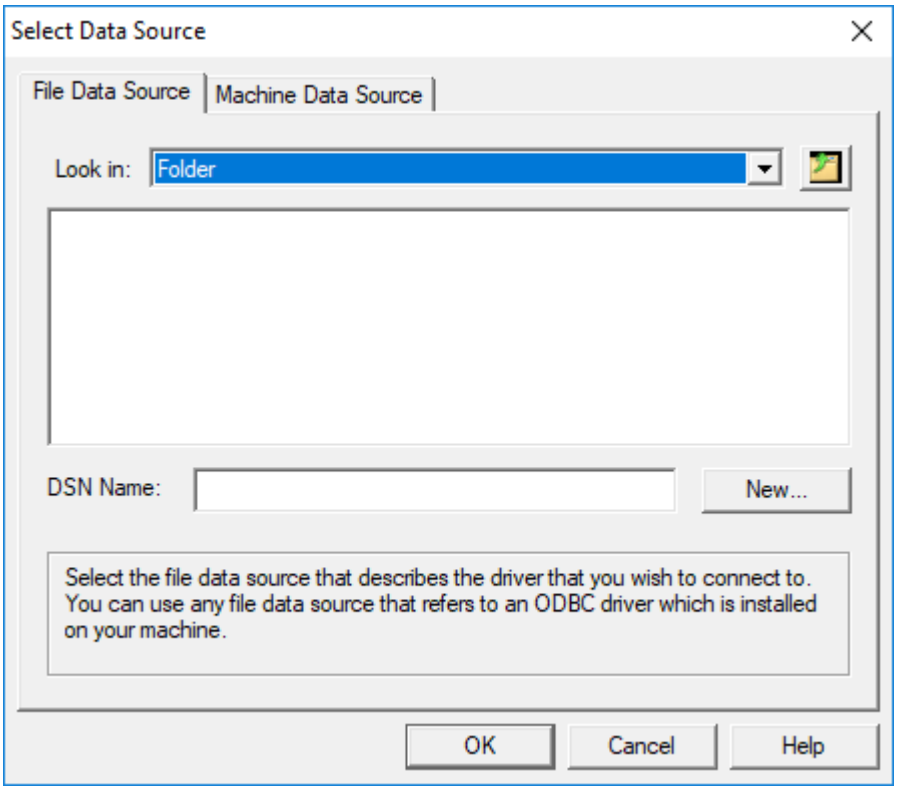
The tool has a graphical user interface and allows you to persist data as an XML data file.

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

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.

Packet logging

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

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

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

To enable TCP packet logging:

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

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

Configuring and enabling packet logging

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

Driver setup dialog (Windows)

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

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

odbc.ini file (UNIX/Linux)

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

```
[Oracle Wire Protocol]
Driver=ODBCHOME/lib/XXora28.so
Description=DataDirect 8.0 Oracle Wire Protocol
...
EnablePacketLogging=1
...
HostName=YourOracleServer
...
LogonID=JOHN
...
PacketLoggingFilePrefix=/tmp/myPacketLog
...
PacketLoggingMaxFileSize=102400
...
PacketLoggingMaxNumFiles=10
...
Password=secret
...
PortNumber=1521
...
ServiceName=sales.us.acme.com
...
```

Connection string

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

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol;HostName=YourOracleServer;
PortNumber=1521;ServiceName=sales.us.acme.comLogonID=JOHN;Password=secret;
EnablePacketLogging=1;PacketLoggingFilePrefix=C:\temp\myPacketLog;
```

Packet logging connection options

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

Table 10: 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.

4

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.

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

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

Table 11: Oracle Wire Protocol Attribute Names

Attribute (Short Name)	Default
AccountingInfo (AI)	None
Action (ACT)	None
AlternateServers (ASRV)	None
ApplicationName (AN)	None

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	60000
AuthenticationMethod (AM)	1 (Encrypt Password)
BatchFailureReturnsError (BFRE)	0 (Disabled)
BindParamsAsUnicode (BPAU)	0 (Disabled)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadBatchSize (BLBS)	1024
BulkLoadFieldDelimiter (BLFD)	, (comma)
BulkLoadOptions (BLO)	0
BulkLoadRecordDelimiter (BLRD)	None
CachedCursorLimit (CCL)	32
CachedDescriptionLimit (CDL)	0
CatalogIncludesSynonyms (CIS)	1 (Enabled)
CatalogOptions (CO)	0 (Disabled)
ClientHostName (CHN)	None
ClientID (CID)	None
ClientUser (CU)	None
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CredentialsWalletEntry (CWE)	None
CredentialsWalletPassword (CWPWD)	None
CredentialsWalletPath (CWPATH)	None
CryptoProtocolVersion (CPV)	TLSv1.3,TLSv1.2
CryptoLibName (CLN)	Empty string

Attribute (Short Name)	Default
DataIntegrityLevel (DIL)	1 (Accepted)
DataIntegrityTypes (DIT)	MD5,SHA1,SHA256,SHA384,SHA512
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
DescribeAtPrepare (DAP)	0 (Disabled)
Description (n/a)	None
EditionName (EN)	None
EnableBulkLoad (EBL)	0 (Disabled)
EnableDescribeParam (EDP)	0 (Disabled)
EnableFIPS (EF)	0 (Default provider)
EnableNcharSupport (ENS) **Note:** EnableNcharSupport has been deprecated.	None
EnableScrollableCursors (ESC)	1 (Enabled)
EnableServerResultCache (ESRC)	0 (Disabled)
EnableStaticCursorsForLongData (ESCLD)	0 (Disabled)
EnableTimestampwithTimezone (ETWT) **Note:** EnableTimestampwithTimezone has been deprecated.	None
EncryptionLevel (EL)	1 (Accepted)
EncryptionMethod (EM)	0 (No Encryption)
EncryptionTypes (ET)	No encryption methods are specified. The driver sends a list of all of the encryption methods to the Oracle server.
Entra Access Token (EAT)	None
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)

Attribute (Short Name)	Default
FailoverPreconnect (FP)	0 (Disabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
GSSClient (GSSC)	native
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) (UNIX and Linux only)	4 (ISO 8559-1 Latin-1)
ImpersonateUser (IU)	None
InitializationString (IS)	None
KeepAlive (KA)	0 (Disabled)
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LDAPCryptoProtocolVersion (LDAPCPV)	TLSv1.3,TLSv1.2
LDAPDistinguishedName (LDAPDN)	None
LDAPEncryptionMethod (LDAPEM)	0
LDAPKeyStore (LDAPKS)	None
LDAPTrustStore (LDAPTS)	None
LDAPValidateServerCertificate (LDAPVSC)	1
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LOBPrefetchSize (LPS)	4000
LocalTimezoneOffset (LTZO)	"" (Empty String)
LockTimeout (LTO)	-1
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100

Attribute (Short Name)	Default
MinPoolSize (MNPS)	0
Module (MOD)	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
Pooling (POOL)	0 (Disabled)
PortNumber (PORT)	None
PRNGSeedFile (PSF) (UNIX and Linux only)	/dev/random
PRNGSeedSource (PSS) (UNIX and Linux only)	0 (File)
ProcedureRetResults (PRR)	0 (Disabled)
ProgramID (PID)	None
ProxyHost (PXHN)	Empty string
ProxyMode (PXM)	0 (NONE)
ProxyPassword (PXPW)	Empty string
ProxyPort (PXPT)	0
ProxyUser (PXU)	Empty string
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
ReportRecycleBin (RRB)	0 (Disabled)
SDU Size on page 245	16384
ServerName (SRVR)	None
ServerType (ST)	0 (Server Default)SDUSize (SDU)

Attribute (Short Name)	Default
ServiceName (SN)	None. If no value is specified for either the SID, Service Name, or TNSNames option, the driver attempts to connect to the ORCL SID by default.
SID (SID)	None. If no value is specified for either the SID, Service Name, or TNSNames option, the driver attempts to connect to the ORCL SID by default.
SSLLibName (SLN)	Empty string
SupportBinaryXML (SBX)	0 (Disabled)
TimestampEscapeMapping (TEM)	0SDUSize (Oracle Version Specific)
TNSNamesFile (TNF)	None. If no value is specified for either the SID, Service Name, or TNSNames option, the driver attempts to connect to the ORCL SID by default.
Truststore (TS)	None
TruststorePassword (TSP)	None
UseCurrentSchema (UCS)	1 (Enabled)
ValidateServerCertificate (VSC)	1 (Enabled)
WireProtocolMode (WPM)	2

For details, see the following topics:

- Accounting Info
- Action
- Alternate Servers
- Application Name
- Application Using Threads
- Array Size
- Authentication Method
- Batch Size
- Batch Failure Returns Error
- Bind Params As Unicode
- Bulk Binary Threshold

- Bulk Character Threshold
- Bulk Options
- Cached Cursor Limit
- Cached Description Limit
- Catalog Functions Include Synonyms
- Catalog Options
- Client Host Name
- Client ID
- Client User
- Connection Pooling
- Connection Reset
- Connection Retry Count
- Connection Retry Delay
- Credentials Wallet Entry
- Credentials Wallet Path
- Crypto Protocol Version
- CryptoLibName
- Data Integrity Level
- Data Integrity Types
- Data Source Name
- Default Buffer Size for Long/LOB Columns (in Kb)
- Describe at Prepare
- Description
- Edition Name
- Enable Bulk Load
- Enable FIPS
- Enable N-CHAR Support
- Enable Scrollable Cursors
- Enable Server Result Cache
- Enable SQLDescribeParam
- Enable Static Cursors for Long Data
- Enable Timestamp with Timezone
- Encryption Level

- Encryption Method
- Entra Access Token
- Encryption Types
- Failover Granularity
- Failover Mode
- Failover Preconnect
- Fetch TSWTZ as Timestamp
- Field Delimiter
- GSS Client Library
- Host
- Host Name In Certificate
- IANAAppCodePage
- Impersonate User
- Initialization String
- Key Password
- Key Store
- Key Store Password
- LDAP Crypto Protocol Version
- LDAP Distinguished Name
- LDAP Encryption Method
- LDAP Key Store
- LDAP Trust Store
- LDAP Validate Server Certificate
- Load Balancing
- LoadBalance Timeout
- LOB Prefetch Size
- Local Timezone Offset
- Lock Timeout
- Login Timeout
- Max Pool Size
- Min Pool Size
- Module
- OpenSSLConfigFile

- OpenSSLProviderPath
- Password
- Port Number
- Proxy Host
- Proxy Mode
- Proxy Password
- Proxy Port
- Proxy User
- PRNGSeedFile
- PRNGSeedSource
- Procedure Returns Results
- Program ID
- Query Timeout
- Record Delimiter
- Report Codepage Conversion Errors
- Report Recycle Bin
- SDU Size
- Server Name
- Server Process Type
- Service Name
- SID
- SSLLibName
- Support Binary XML
- TCP Keep Alive
- Timestamp Escape Mapping
- TNSNames File
- Trust Store
- Trust Store Password
- Use Current Schema for SQLProcedures
- User Name
- Validate Server Certificate
- Wallet Password
- Wire Protocol Mode

Accounting Info

Attribute

AccountingInfo (AI)

Purpose

Accounting information to be stored in the database. This value sets the CLIENT_INFO value of the V$SESSION table on the server. This value is used by the client information feature.

Valid Values

string

where:

string

is the accounting information.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Action

Attribute

Action (ACT)

Purpose

The current action (Select, Insert, Update, or Delete, for example) within the current module. This value sets the ACTION column of the V$SESSION table on the server. This value is used by the client information feature.

This option only applies to connections to Oracle 10*g* R2 and higher database servers.

Valid Values

string

where:

string

is the current action.

Notes

- You can also specify this information using the Oracle DBMS_APPLICATION_INFO.SET_ACTION procedure or the DBMS_APPLICATION_INFO.SET_MODULE procedure.
- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

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:{SID=sidvalue | ServiceName=servicevalue}[,
 . . .])
```

You must specify the host name, port number, and either the SID or service name of each alternate server.

Example

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

```
(HostName=AccountingOracleServer:PortNumber=1521:
SID=Accounting,HostName=255.201.11.24:PortNumber=1522:
ServiceName=ABackup.NA.MyCompany)
```

Default

None

GUI tab

Failover tab

Application Name

Attribute

ApplicationName (AN)

Purpose

The name of the application to be stored in the database. This value sets the dbms_session value in the database and the PROGRAM value of the V$SESSION table on the server. This value is used by the client information feature.

Valid Values

string

where:

string

is the name of the application.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Application Using Threads

Attribute

ApplicationUsingThreads (AUT)

Purpose

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

Valid Values

0 | 1

Behavior

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

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

Notes

- This connection option can affect performance.

Default

1 (Enabled)

GUI tab

Advanced tab

See also

Performance considerations on page 92

Array Size

Attribute

ArraySize (AS)

Purpose

The number of bytes the driver can fetch in a single network round trip. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.

Valid Values

x

where

x

is a positive integer. The value 1 does not define the number of bytes, but, instead, causes the driver to allocate space for exactly one row of data.

Notes

- This connection option can affect performance.
- The setting of the Array Size property can be overridden by specifying the number of rows to fetch using the SQL_ATTR_ROW_ARRAY_SIZE statement attribute. When issuing the statement attribute, the driver calculates the internal buffer size for a fetch by multiplying the number of rows specified by the row size

reported in the server metadata. Therefore, specifying large values may improve throughput, but it does so at the expense of increased demands on memory.

Default

60000

GUI Tab

Performance tab

See also

Performance considerations on page 92

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

1 | 3 | 4 | 5 | 6 | 11 | 12 | 16 | 38

Behavior

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

If set to 3 (Client Authentication), the driver uses client authentication when establishing a connection. The database server relies on the client to authenticate the user and does not provide additional authentication.

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

When set to 5 (Kerberos with UID & PWD), the driver uses both Kerberos authentication and user ID and password authentication. The driver first authenticates the user using Kerberos. If a user ID and password are specified, the driver reauthenticates using the user name and password supplied. An error is generated if a user ID and password are not specified.

If set to 6 (NTLM), the driver uses NTLMv1 authentication for Windows clients.

If set to 11 (SSL), the driver uses SSL certificate information to authenticate the client with the server when using Oracle Wallet. The User Name and Password options should **not** be specified. See "Oracle Wallet SSL Authentication" for additional requirements.

If set to 12 (SSL with UID & Password), the driver uses user ID, password and SSL authentication to connect with the server when using Oracle Wallet. See "Oracle Wallet SSL Authentication" for additional requirements.

If set to 16 (Wallet UID & PWD), the driver authenticates to the server using a user ID and password retrieved from Oracle Wallet. See "Oracle Wallet Password Store" for additional requirements.

If set to 38 (EntraIDAccessToken), the driver authenticates to the server using an Entra ID access token. This setting requires the Entra Access Token (EntraAccessToken) option to be specified. If an access token is not specified, the driver throws an exception. All communications between the service are encrypted using TLS/SSL encryption.

Notes

- When AuthenticationMethod is set to 16 (Wallet UID & PWD), specifying values for the User Name (LogonID) or Password (Password) options returns a warning and the values are ignored.

Default

1 (Encrypt Password)

GUI tab

Security tab

See Also

- User Name on page 256
- Password on page 232
- Oracle Wallet SSL Authentication on page 121
- Oracle Wallet Password Store on page 122
- Authentication on page 119

Batch Size

Supported on Windows, UNIX, and Linux only.

Attribute

BulkLoadBatchSize (BLBS)

Purpose

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

Valid Values

0 | *x*

where

x

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

Default

1024

GUI Tab

Bulk tab

Batch Failure Returns Error

Attribute

BatchFailureReturnsError (BFRE)

Purpose

Determines the behavior of the driver when encountering an error in a parameter array insert with bulk load disabled (EnableBulkLoad=0). Note that this option applies only when operating in autocommit mode.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), the driver returns SQL_ERROR and rolls back the operation when encountering an error in any of the parameter sets.

If set to 0 (Disabled), the driver returns SQL_SUCCESS_WITH_INFO and commits the rows that were successfully inserted prior to encountering the error.

Default

0 (Disabled)

GUI Tab

Advanced tab

Bind Params As Unicode

Attribute

BindParamsAsUnicode (BPAU)

Purpose

Specifies whether the driver converts data in bind parameters from the SQL_CHAR, SQL_VARCHAR and SQL_LONGVARCHAR ODBC data types to the SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR types when C type is set to SQL_C_WCHAR. When certain applications bind SQL_C_WCHAR data to a non-Unicode ODBC type, this behavior may result in the substitution of some characters. Enabling this option allows you to avoid character substitution by configuring the driver to use the corresponding Unicode ODBC type.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver converts data bound to the SQL_CHAR, SQL_VARCHAR and SQL_LONGVARCHAR ODBC data types to the SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCAR data types when C type is set to SQL_C_WCHAR.

If set to `0` (Disabled), the driver does not convert data bound to the SQL_CHAR, SQL_VARCHAR and SQL_LONGVARCHAR data types.

Default

`0` (Disabled)

GUI Tab

Advanced Tab

Bulk Binary Threshold

Supported on Windows, UNIX, and Linux only.

Attribute

BulkBinaryThreshold (BBT)

Purpose

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

Valid Values

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

where

x

is an integer that specifies the number of KB.

Behavior

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

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

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

Default

32

GUI Tab

Bulk tab

Bulk Character Threshold

Supported on Windows, UNIX, and Linux only.

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

Bulk Options

Supported on Windows, UNIX, and Linux only.

Attribute

BulkLoadOptions (BLO)

Purpose

Toggles options for the bulk load process.

This option only applies to connections to Oracle 11g R2 and higher database servers.

Valid Values

`0` | *x*

where:

x

is a positive integer representing the cumulative total of the Bulk Options values.

Behavior

If set to `0`, none of the options for bulk load are enabled.

If set to *x*, the values represented by x are enabled.

Currently, the only bulk load option available is:

No Index Errors - The driver stops a bulk load operation when a value that would cause an index to be invalidated is loaded. For example, if a value is loaded that violates a unique or non-null constraint, the driver stops the bulk load operation and discards all data being loaded, including any data that was loaded prior to the problem value. If not enabled, the bulk load operation continues even if a value that would cause an index to be invalidated is loaded. Value=`128`.

Notes

- The cumulative value of the options is only used in a connection string with the connection string attribute, BulkLoadOptions. On the Bulk tab of the driver Setup dialog, the individual options are enabled by selecting the appropriate check box.

Default

`0` (disabled)

GUI Tab

Bulk tab

Cached Cursor Limit

Attribute

CachedCursorLimit (CCL)

Purpose

Specifies the number of Oracle Cursor Identifiers that the driver stores in cache. A Cursor Identifier is needed for each concurrent open Select statement. When a Select statement is closed, the driver stores the identifier in its cache, up to the limit specified, rather than closing the Cursor Identifier. When a new Cursor Identifier is needed, the driver takes one from its cache, if one is available. Cached Cursor Identifiers are closed when the connection is closed.

Valid Values

An integer from `0` to `65535`

Default

`32`

GUI Tab

Performance tab

See also

Performance considerations on page 92

Cached Description Limit

Attribute

CachedDescriptionLimit (CDL)

Purpose

Specifies the number of descriptions that the driver saves for Select statements. These descriptions include the number of columns, data type, length, and scale for each column. The matching is done by an exact-text match through the FROM clause.

Valid Values

An integer from `0` to `65535`

Notes

- If the Select statement contains a Union or a nested Select, the description is not cached.

Default

0

GUI Tab

Performance tab

See also

Performance considerations on page 92

Catalog Functions Include Synonyms

Attribute

CatalogIncludesSynonyms (CIS)

Purpose

Determines whether synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.

Valid Values

0 | 1

Behavior

If set to 1 (Enabled), synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.

If set to 0 (Disabled), synonyms are excluded (a non-standard behavior) and performance is thereby improved.

Notes

- This connection option can affect performance.

Default

1 (Enabled)

GUI Tab

Performance tab

See also

Performance considerations on page 92

Catalog Options

Attribute

CatalogOptions (CO)

Purpose

Determines whether SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the result column REMARKS (for the catalog functions SQLTables and SQLColumns) and the result column COLUMN_DEF (for the catalog function SQLColumns) return actual values. Enabling this option reduces the performance of your catalog (SQLColumns and SQLTables) queries.

If set to `0` (Disabled), SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.

Notes

- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Advanced tab

See also

Performance considerations on page 92

Client Host Name

Attribute

ClientHostName (CHN)

Purpose

The host name of the client machine to be stored in the database. This value sets the MACHINE value in the V$SESSION table on the server. This value is used by the client information feature.

Valid Values

`string`

where:

string

is the host name of the client machine.

If a value for this option is not specified, the driver uses the current machine name and IP address in the following format:

machine_name/IP_address

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Client ID

Attribute

ClientID (CID)

Purpose

Additional information about the client to be stored in the database. This value sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server. This value is used by the client information feature.

This option only applies to connections to Oracle 10*g* R2 and higher database servers.

Valid Values

string

where:

string

is additional information about the client.

Notes

- You can also specify this information using the Oracle DBMS_SESSION.SETIDENTIFIER procedure or the DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure.
- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Client User

Attribute

ClientUser (CU)

Purpose

The user ID to be stored in the database. This value sets the OSUSER value in the V$SESSION table on the server. This value is used by the client information feature.

Valid Values

`-1` | *`string`*

where:

`string`

is a valid user ID.

Behavior

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

If a value for this option is not specified, the driver uses name of the user that is currently logged into the OS.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Connection Pooling

Supported on Windows, UNIX, and Linux only.

Attribute

Pooling (POOL)

Purpose

Specifies whether to use the driver's connection pooling.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver uses connection pooling.

If set to `0` (Disabled), the driver does not use connection pooling.

Notes

- The application must be thread-enabled to use connection pooling.
- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Pooling tab

See also

Performance considerations on page 92

Connection Reset

Supported on Windows, UNIX, and Linux only.

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 92

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

Credentials Wallet Entry

Attribute

CredentialsWalletEntry (CWE)

Purpose

Specifies the string value used to identify database credential information stored in an Oracle Wallet. When `AuthenticationMethod=16` (Wallet UID & PWD), the driver retrieves the user ID and password associated with the specified value from the wallet and uses them to authenticate to the server. This value provides a method for the correct user ID and password to be retrieved when there are multiple pairs in a wallet.

See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.

Valid Values

`wallet_entry_string`

where:

`wallet_entry_string`

the string used to identify sets of database credential information stored in a wallet. This value is defined when creating or modifying credentials stored in a wallet and is typically a net service name, Oracle service name, or *`host`*:*`port`*:*`SID`* string, but can be any value specified by the user creating the credentials entry.

Notes

- When `AuthenticationMethod=16` (Wallet UID & PWD), the driver retrieves user ID and passwords from the Oracle Wallet file specified by the Credentials Wallet Path (CredentialsWalletPath) option.

Default

None

GUI Tab

Security tab

See Also

- Authentication Method on page 176
- Wallet Password on page 257
- Oracle Wallet Password Store on page 122

Credentials Wallet Path

Attribute

CredentialsWalletPath (CWPATH)

Purpose

Specifies the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. When `AuthenticationMethod=16` (Wallet UID & PWD), the driver retrieves the database user name and password from this file.

See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.

Valid Values

`wallet_file_path`

where:

`wallet_file_path`

the fully-qualified path to the Oracle Wallet file in which your database credential information is stored.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the Credentials Wallet Path option. The Credentials Wallet Path option provides a method for you to specify a Oracle Wallet file used to authenticate to your database. 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.
- An Oracle Wallet can contain multiple User ID password pairs. To ensure the driver retrieves the correct credentials, you must specify the identifier string for the credentials using the Credentials Wallet Entry (CredentialsWalletEntry) option.
- If you are using an `ewallet.p12` file for your wallet, specify the Oracle Wallet file password using the Wallet Password (CredentialsWalletPassword) option.

Default

None

GUI Tab

Security tab

See Also

- Authentication Method on page 176
- Wallet Password on page 257
- Oracle Wallet Password Store on page 122

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.3 | TLSv1.2

Example

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

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

Notes

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

Default

TLSv1.3,TLSv1.2

GUI Tab

Security tab

CryptoLibName

Attribute

CryptoLibName (CLN)

Purpose

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

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

Valid Values

*absolute_path**openssl_filename*

where:

absolute_path

is the absolute path to where the OpenSSL file is located

openssl_filename

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

Example

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

Notes

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

Default

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 also

- SSLLibName on page 249
- Advanced tab on page 53

Data Integrity Level

Attribute

DataIntegrityLevel (DIL)

Purpose

Specifies a preference for the data integrity to be used on data sent between the driver and the database server. The connection fails if the database server does not have a compatible integrity algorithm. See "Encryption and data integrity" for more information.

Valid Values

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

Behavior

If set to `0` (Rejected), a data integrity check on data sent between the driver and the database server is refused. The connection fails if the database server specifies REQUIRED.

If set to `1` (Accepted), a data integrity check can be made on data sent between the driver and the database server. Data integrity is used if the database server requests or requires it.

If set to `2` (Requested), the driver enables a data integrity check on data sent between the driver and the database server if the database server permits it.

If set to `3` (Required), a data integrity check must be performed on data sent between the driver and the database server. The connection fails if the database server specifies REJECTED.

Notes

- Consult your database administrator concerning the data integrity settings of your Oracle server.
- This connection option can affect performance.

Default

`1` (Accepted)

GUI Tab

Advanced Security tab

See also

- Oracle Advanced Security on page 135
- Performance considerations on page 92

Data Integrity Types

Attribute

DataIntegrityTypes (DIT)

Purpose

Determines the method the driver uses to protect against attacks that intercept and modify data being transmitted between the client and server. You can enable data integrity protection without enabling encryption. See "Encryption and data integrity" for more information.

Valid Values

value [[, *value*]...]

where:

value

is one of the following cryptographic algorithms:

MD5 | SHA1 | SHA256 | SHA384 | SHA512

If multiple values are specified and Oracle Advanced Security data integrity is enabled using the Data Integrity Level option, the database server determines which algorithm is used based on how it is configured.

Notes

- This option has no effect if "Data Integrity Level" is set to 0 - Rejected.
- Consult your database administrator concerning the data integrity settings of your Oracle server.
- This connection option can affect performance.

Default

MD5,SHA1,SHA256,SHA384,SHA512

GUI Tab

Advanced Security tab

See also

- Data Integrity Level on page 194
- Oracle Advanced Security on page 135
- Performance considerations on page 92

Data Source Name

Attribute

DataSourceName (DSN)

Description

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

Valid Values

`string`

where:

`string`

is the name of a data source.

Default

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.

Valid Values

An integer in multiples of `1024`

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

Notes

- This connection option can affect performance.

Default

1024

GUI Tab

Advanced tab

See also

Performance considerations on page 92

Describe at Prepare

Attribute

DescribeAtPrepare (DAP)

Purpose

Determines whether the driver describes the SQL statement at prepare time.

This connection option can affect performance.

Valid Values

0 | 1

If set to 1 (Enabled), the driver describes the SQL statement at prepare time.

If set to 0 (Disabled), the driver does not describe the SQL statement at prepare time.

Notes

- This connection option can affect performance.

Default

0 (Disabled)

GUI Tab

Advanced tab

See also

Performance considerations on page 92

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

Edition Name

Attribute

EditionName (EN)

Purpose

The name of the Oracle edition the driver uses when establishing a connection. Oracle 11*g* R2 and higher allows your database administrator to create multiple editions of schema objects so that your application can still use those objects while the database is being upgraded. This option is only valid for Oracle 11*g* R2 and higher databases and tells the driver which edition of the schema objects to use.

The driver uses the default edition in the following cases:

- When the specified edition is not a valid edition. The driver generates a warning indicating that it was unable to set the current edition to the specified edition.
- When the value for this option is not specified or is set to an empty string.

If failover is enabled using the Failover Mode connection option and a connection fails over to another database server, the driver connects to the alternate server using the same edition that was used for the failed connection. The driver does not track changes to the current edition made using the ALTER SESSION SQL statement.

Valid Values

`string`

where:

`string`

is the name of a valid Oracle edition.

Default

None

GUI Tab

General tab

Enable Bulk Load

Supported on Windows, UNIX, and Linux only.

Attribute

EnableBulkLoad (EBL)

Purpose

Specifies the bulk load method.

To override the value set by this connection option for an individual statement, set a different value in the SQL_ATTR_BULK_LOAD_ENABLED statement attribute (attribute value 1062) on the SQLSetStmtAttr() function. Setting the attribute has the same behavior as setting this connection option. To enable this option, set a value of SQL_TRUE. To disable, set a value of SQL_FALSE. This statement is defined in the `qesqlext.h` file, which is installed with the driver.

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.

Statement Attribute

To override the value set by this connection option for an individual statement, set a different value in the SQL_ATTR_BULK_LOAD_ENABLED statement attribute on the SQLSetStmtAttr() function. Specify one of the following values when using

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

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

Default

`0` (Disabled)

GUI Tab

Bulk Tab

See Also

Performance considerations on page 92

Enable FIPS

Attribute

EnableFIPS (EF)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Notes

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

Default

`0`

See also

TLS/SSL server authentication on page 125
TLS/SSL client authentication on page 130
Generating TLS/SSL certificates using OpenSSL 3.5-compliant algorithms on page 132

Enable N-CHAR Support

Note: The Enable N-CHAR Support connection option has been deprecated, and the driver behavior has been updated to always provide support for the N-types NCHAR, NVARCHAR2 and NCLOB. For compatibility purposes, the EnableNcharSupport attribute can still be manually configured for this release, but will be deprecated in subsequent versions of the product. To configure the attribute on Windows, use the Extended Options field on the Advanced tab. For UNIX/Linux, using a text editor, add the attribute to your data source in the `odbc.ini` file.

Attribute

EnableNcharSupport (ENS)

Purpose

Determines whether the driver provides support for the N-types NCHAR, NVARCHAR2, and NCLOB. These types are described as SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR, and are returned as supported by SQLGetTypeInfo. In addition, the "normal" char types (char, varchar2, long, clob) are described as SQL_CHAR, SQL_VARCHAR, and SQL_LONGVARCHAR regardless of the character set on the Oracle server.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver provides support for the N-types NCHAR, NVARCHAR2, and NCLOB.

If set to `0` (Disabled), the driver does not provide support for the N-types NCHAR, NVARCHAR2, and NCLOB.

Notes

- Valid only on Oracle 9*i* and higher.

Default

None

See also

- Advanced tab on page 53

GUI Tab

None

Enable Scrollable Cursors

Attribute

EnableScrollableCursors (ESC)

Purpose

Determines whether scrollable cursors, both Keyset and Static, are enabled for the data source.

This connection option can affect performance.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), scrollable cursors are enabled for the data source.

If set to `0` (Disabled), scrollable cursors are not enabled.

Notes

- This connection option can affect performance.

Default

`1` (Enabled)

GUI Tab

Performance tab

See also

Performance considerations on page 92

Enable Server Result Cache

Attribute

EnableServerResultCache (ESRC)

Purpose

Determines whether the driver sets the RESULT_CACHE_MODE session parameter to FORCE.

This option only applies to connections to Oracle 11*g* or higher database servers that support server-side result set caching.

This connection option can affect performance.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver sets the RESULT_CACHE_MODE session parameter to FORCE.

If set to `0` (Disabled), the driver does not sets the RESULT_CACHE_MODE session parameter.

Notes

- Oracle Autonomous Data Warehouse Cloud requires server-side result caching to be enabled. Therefore, this property is ignored and server-side result caching is always enabled when connected to the Oracle Autonomous Data Warehouse service.
- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Advanced tab

See also

Performance considerations on page 92

Enable SQLDescribeParam

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. 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 and returns the error: `unimplemented function`.

Default

`0` (Disabled)

GUI Tab

Advanced tab

Enable Static Cursors for Long Data

Attribute

EnableStaticCursorsForLongData (ESCLD)

Purpose

Determines whether the driver supports Long columns when using a static cursor. Enabling this option causes a performance penalty at the time of execution when reading Long data.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver supports Long columns when using a static cursor.

If set to `0` (Disabled), the driver does not support Long columns when using a static cursor.

Notes

- You must enable this option if you want to persist a result set that contains Long data into an XML data file.
- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Performance tab

See also

Performance considerations on page 92

Enable Timestamp with Timezone

Note: The Enable Timestamp with Timezone connection has been deprecated, and the driver behavior has been updated to always expose timestamps with timezones to the application. For compatibility purposes, the EnableTimestampwithTimezone attribute can still be manually configured for this release, but will be deprecated in subsequent versions of the product. To configure the attribute on Windows, use the Extended Options field on the Advanced tab. For UNIX/Linux, using a text editor, add the attribute to your data source in the `odbc.ini` file.

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. The driver issues an ALTER SESSION at connection time to modify NLS_TIMESTAMP_TZ_FORMAT. NLS_TIMESTAMP_TZ_FORMAT is changed to the ODBC definition of a timestamp literal with the addition of the timezone literal: '*YYYY-MM-DD HH24:MI:SSXFF TZR*'.

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

Default

None

See Also

- Advanced tab on page 53

GUI Tab

None

Encryption Level

Attribute

EncryptionLevel (EL)

Purpose

Specifies a preference on whether to use encryption on data being sent between the driver and the database server.

Valid Values

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

Behavior

If set to `0` (Rejected), or if no match is found between the driver and server encryption types, data sent between the driver and the database server is not encrypted or decrypted. The connection fails if the database server specifies REQUIRED.

If set to `1` (Accepted), encryption is used on data sent between the driver and the database server if the database server requests or requires it.

If set to 2 (Requested), data sent between the driver and the database server is encrypted and decrypted if the database server permits it.

If set to 3 (Required), data sent between the driver and the database server must be encrypted and decrypted. The connection fails if the database server specifies REJECTED.

Notes

- Consult your database administrator concerning the data encryption settings of your Oracle server.
- This connection option can affect performance.

Default

1 (Accepted)

GUI Tab

Advanced Security tab

See also

Performance considerations on page 92

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 TLS/SSL protocols specified in the Crypto Protocol Version connection option.

Notes

- Consult your database administrator concerning the TLS/SSL settings of your Oracle server.
- This connection option can affect performance.
- When using FIPS and default providers, the certificates must be generated using the OpenSSL 3.5-compliant cryptographic algorithms.

Default

0 (No Encryption)

GUI Tab

Security tab

See also

Performance considerations on page 92

Entra Access Token

Attribute

EntraAccessToken (EAT)

Purpose

Specifies the access token required to authenticate to an Oracle instance when using Entra ID access token authentication (`AuthenticationMethod=38`). Refer to the Oracle documentation for more information on obtaining an access token.

Valid Values

access_token

where:

access_token

is an Entra ID access token. When using a connection string, the value of this option should be enclosed in double quotes to avoid potential issues with special characters in the token.

Notes

- When using Entra ID access token authentication, the driver encrypts data using TLS/SSL encryption; therefore, the Encryption Method (EncryptionMethod) option is automatically set to 1 (SSL) when Entra ID authentication is enabled.

Default

No default value

GUI Tab

Logon dialog

See Also

- Authentication Method on page 176
- Encryption Method on page 206
- Entra ID access token authentication on page 123

Encryption Types

Attribute

EncryptionTypes (ET)

Purpose

Specifies a comma-separated list of the encryption algorithms to use if Oracle Advanced Security encryption is enabled using the Encryption Level connection property.

Valid Values

encryption_algorithm [[, *encryption_algorithm*]...]

where:

encryption_algorithm

is a encryption algorithm specifying an algorithm in the following table:

AES256 | RC4_256 | AES192 | 3DES168 | AES128 | RC4_128 | 3DES112 | RC4_56 | DES | RC4_40

Encryption Algorithm	Description
3DES112	Two-key Triple-DES (with an effective key size of 112-bit).
AES128	AES with a 128-bit key size.
AES192	AES with a 192-bit key size.
AES256	AES with a 256-bit key size.
DES	DES (with an effective key size of 56-bit).
DES168	Three-key Triple-DES (with an effective key size of 168-bit).
RC4_128	RC4-128 with a 128-bit key size.
RC4_256	RC4 with a 256-bit key size.
RC4_40	RSA RC4 with a 40-bit key size.
RC4_56	RSA RC4 with a 56-bit key size.

Example

Your security environments specifies that you can use RC4 with a 256-bit key size, AES with a 192-bit key size, or two-key Triple-DES with an effective key size of 112-bit. Use the following values:

```
EncryptionTypes=RC4_256,AES192,3DES112
```

Notes

- This option is ignored if Encryption Level is set to `0` (Rejected).
- Consult your database administrator concerning the data encryption settings of your Oracle server.
- This connection option can affect performance.

Default

On the GUI tab: all check boxes are selected.

In the connection string: no encryption methods are specified. The driver sends a list of all of the encryption methods to the Oracle server.

GUI Tab

Advanced Security tab

See also

Performance considerations on page 92

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)

Description

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 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 on Oracle 10*g* R2 or higher.

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

Field Delimiter

Supported on Windows, UNIX, and Linux only.

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

, (comma)

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 `native`, the driver uses the GSS client shipped with the operating system.

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

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

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, or a combination of the two.

Notes

- This option is mutually exclusive with the Server Name and TNSNames File options.
- When a value is specified for the LDAP Distinguished Name option, this option specifies the name or IP address of the LDAP directory server.

Default

None

GUI Tab

General tab

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when TLS/SSL encryption is enabled (`EncryptionMethod= 1 | 3 | 4 | 5`) and validation is enabled (`ValidateServerCertificate=1`). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values

host_name | *#SERVERNAME#*

where

host_name

is the host name specified in the certificate. Consult your TLS/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

UNIX®

Supported on UNIX and Linux only.

Attribute

IANAAppCodePage (IACP)

Purpose

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

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

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

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

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

To override the value set by this connection option for an individual statement, set a different value in the SQL_ATTR_IANA_APP_CODE_PAGE statement attribute (attribute value 1064) on the SQLSetStmtAttr() function. This statement is defined in the `qesqlext.h` file, which is installed with the driver.

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

- If your database is set to Unicode, specifying a value of `106` passes Unicode data to a your database without conversion. This value eliminates the need for ODBC function calls.

Default

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

GUI Tab

Advanced tab

See Also

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

Impersonate User

Attribute

ImpersonateUser (IU)

Purpose

Specifies the proxy user ID used for impersonation. The value for Impersonate User determines your identity and permissions when executing queries. When a value is specified for this option, the driver authenticates according to the setting of the Authentication Method option; then, after establishing a connection, the driver attempts to reauthenticate as the destination user. Note that the administrator must grant CONNECT THROUGH permission to the authenticated user in order to impersonate the destination user; otherwise, an error is returned.

Valid Values

`destination_userid`

where:

`destination_userid`

is a valid user ID with permissions to access the database. Case-sensitive values must be enclosed in either single or double quotation marks.

Default

None

GUI Tab

Security 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:

```
Initialization String=ALTER SESSION SET DATE_FORMAT = 'DD/MM/YYYY'
```

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

Key Password

Attribute

KeyPassword (KP)

Purpose

The password used to access the individual keys in the keystore file when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/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 absolute path to the keystore file to be used when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/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

path

where:

path

is the absolute path to the keystore file.

Notes

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

Default

None

GUI Tab

Security tab

Key Store Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when TLS/SSL is enabled (`EncryptionMethod=1`) and TLS/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

LDAP Crypto Protocol Version

Attribute

LDAPCryptoProtocolVersion (LDAPCPV)

Purpose

Specifies a cryptographic protocol or comma-separated list of cryptographic protocols that can be used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`).

Valid Values

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

where:

cryptographic_protocol

is one of the following cryptographic protocols:

TLSv1.3 | TLSv1.2 |

Example

If your server supports TLSv1.3 and TLSv1.2, you can specify acceptable cryptographic protocols with the following key-value pair:

```
LDAPCryptoProtocolVersion=TLSv1.3,TLSv1.2
```

Notes

- 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 server, the connection fails and the driver returns an error.

Default

TLSv1.3,TLSv1.2

GUI Tab

Advanced Security tab

See Also

- Using LDAP on page 95
- TLS/SSL encryption on page 124

LDAP Distinguished Name

Supported on Windows, UNIX, and Linux only.

Attribute

LDAPDistinguishedName (LDAPDN)

Purpose

Specifies the distinguished name for the LDAP entry that contains your connection information. Using an LDAP entry provides simplified maintenance by allowing you to centrally store and access connection information. LDAP entries specify the Host, Port Number, and Service Name or SID for the target database using the orclNetDescString attribute.

Valid Values

`distinguished_name`

where:

`distinguished_name`

is the fully qualified path of names in the LDAP directory information tree for the entry containing your connection information. For example, `cn=DB122,cn=OracleContext,dc=america,dc=yourcompany,dc=com`.

Notes

- This option is mutually exclusive with the TNSNames File (TNSNamesFile), SID (SID), and Service Name (ServiceName) options.
- If a value is specified for this option, the Host (HostName) and Port Number (PortNumber) options are used to specify the host and port number for the LDAP directory server.

Default

No default value

GUI Tab

General tab

See Also

- Host on page 213
- Port Number on page 233
- Using LDAP on page 95

LDAP Encryption Method

Attribute

LDAPEncryptionMethod (LDAPEM)

Purpose

Determines whether data is encrypted and decrypted when transmitted over the network between the driver and LDAP server.

Valid Values

`0 | 1`

Behavior

If set to `0` (No Encryption), data is neither encrypted nor decrypted.

If set to `1` (SSL), data is encrypted using TLS/SSL. If the LDAP server does not support TLS/SSL, the connection fails and the driver throws an exception.

Notes

- Connection hangs can occur when the driver is configured for TLS/SSL and the LDAP server does not support TLS/SSL.

Default

No default value

GUI Tab

Advanced Security tab

See Also

- Using LDAP on page 95
- TLS/SSL encryption on page 124

LDAP Key Store

Attribute

LDAPKeyStore (LDAPKS)

Purpose

Specifies the absolute path to the keystore file used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). The keystore file contains certificates that the client presents to the LDAP server in response to the LDAP server's certificate request.

Valid values

`string`

where:

`string`

is the absolute path to the keystore file.

Notes

- The driver supports only the `.pem` file format for the keystore files.
- For more information on keystores and TLS/SSL encryption, see "TLS/SSL encryption".

Default

No default value

GUI Tab

Advanced Security tab

See Also

- Using LDAP on page 95
- TLS/SSL encryption on page 124

LDAP Trust Store

Attribute

LDAPTrustStore (LDAPTS)

Purpose

Specifies the absolute path to the truststore file used when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). The truststore file contains certificates that the client uses to verify the LDAP server's certificate.

Valid values

`string`

where:

`string`

is the absolute path to the truststore file.

Notes

- This property is ignored if ValidateServerCertificate=`0`.
- The driver supports only the `.pem` file format for the keystore files.
- For more information on truststores and TLS/SSL encryption, see "TLS/SSL encryption".

Default

No default value

GUI Tab

Advanced Security tab

See Also

- Using LDAP on page 95
- TLS/SSL encryption on page 124

LDAP Validate Server Certificate

Attribute

LDAPValidateServerCertificate (LDAPVSC)

Purpose

Determines whether the driver validates the certificate sent by the server when TLS/SSL encryption is enabled for connections to the LDAP server (`LDAPEncryptionMethod=1`). When TLS/SSL encryption is used, the server's certificate must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate, even if it is not issued by a trusted CA, can be useful in test environments, as it eliminates the need to configure truststore information on each client.

Valid values

`0 | 1`

Behavior

If set to `1`, the driver validates the certificate that is sent by the LDAP server. Any certificate from the server must be issued by a trusted CA in the truststore file.

If set to `0`, the driver does not validate the certificate that is sent by the LDAP server. The driver ignores any truststore information that is specified by the LDAP Trust Store option.

Default

`1`

GUI Tab

Advanced Security tab

See Also

- Using LDAP on page 95
- TLS/SSL encryption on page 124

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).

Default

`0` (Disabled)

GUI Tab

Failover tab

LoadBalance Timeout

Supported on Windows, UNIX, and Linux only.

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

LOB Prefetch Size

Attribute

LOBPrefetchSize (LPS)

Purpose

Specifies the size of prefetch data the server returns for BLOBs and CLOBs. LOB Prefetch Size is supported for Oracle database versions 12.1.0.1 and higher.

Valid Values

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

where:

x

is a positive integer that represents the size of a BLOB in bytes or the size of a CLOB in characters.

Behavior

If set to `-1`, the property is disabled.

If set to `0`, the server returns only LOB meta-data such as LOB length and chunk size with the LOB locator during a fetch operation.

If set to *x*, the server returns LOB meta-data and the beginning of LOB data with the LOB locator during a fetch operation. This can have significant performance impact, especially for small LOBs which can potentially be entirely prefetched, because the data is available without having to go through the LOB protocol.

Default

`4000`

GUI Tab

Performance tab

See Also

Performance considerations on page 92

Local Timezone Offset

Attribute

LocalTimezoneOffset (LTZO)

Purpose

A value to alter local time zone information. The default is "" (empty string), which means that the driver determines local time zone information from the operating system. If it is not available from the operating system, the driver defaults to using the setting on the Oracle server.

Valid Values

Valid values are specified as offsets from GMT as follows: (-)*HH*:*MM*. For example, *-08:00* equals GMT minus 8 hours.

The driver uses the value of this option to issue an ALTER SESSION for local time zone at connection time.

Default

"" (empty string)

GUI Tab

Advanced tab

Lock Timeout

Attribute

LockTimeout (LTO)

Purpose

Specifies the amount of time, in seconds, the Oracle server waits for a lock to be released before generating an error when processing a Select...For Update statement on an Oracle 9*i* or higher server.

This connection option can affect performance.

Valid Values

-1 | 0 | *x*

where:

x

is an integer that specifies a number of seconds.

Behavior

If set to -1, the server waits indefinitely for the lock to be released.

If set to `0`, the server generates an error immediately and does not wait for the lock to time out.

If set to *x*, the server waits for the specified number of seconds for the lock to be released.

Notes

- This connection option can affect performance.

Default

`-1`

GUI Tab

Performance tab

See also

Performance considerations on page 92

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

Supported on Windows, UNIX, and Linux only.

Attribute

MaxPoolSize (MXPS)

Purpose

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

Valid Values

An integer from 1 to 65535

Notes

- This connection option can affect performance.

Example

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

Default

100

GUI Tab

Pooling tab

See also

Performance considerations on page 92

Min Pool Size

Supported on Windows, UNIX, and Linux only.

Attribute

MinPoolSize (MNPS)

Purpose

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

Valid Values

`0` | *x*

where:

x

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

Behavior

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

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 92

Module

Attribute

Module (MOD)

Purpose

Provides additional information about the client to be stored in the database. This value sets the CLIENT_IDENTIFIER value in the V$SESSION table on the server. This value is used by the client information feature.

This option only applies to connections to Oracle 10*g* R2 and higher database servers.

Valid Values

string

where:

string

is a the name of a stored procedure or the name of the application.

Notes

- If a value is not specified for this option, the driver uses the PROGRAM value in the V$SESSION table.
- You can also specify this information using the Oracle DBMS_SESSION.SETIDENTIFIER procedure or the DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure.
- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

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

Logon dialog

Port Number

Attribute

PortNumber (PORT)

Description

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.

Notes

- This option is mutually exclusive with the Server Name and TNSNames File options.
- When a value is specified for the LDAP Distinguished Name option, this option specifies the port number for the listener of the LDAP directory server.

Default

For Oracle database server: 1522

For LDAP server (unencrypted): 389

For LDAP server (encrypted): 636

GUI Tab

General tab

Proxy Host

Supported on Windows, UNIX, and Linux only.

Attribute

ProxyHost (PXHN)

Purpose

Specifies the Hostname and possibly the Domain of the Proxy Server. The value specified can be a host name, a fully qualified domain name, or an IPv4 or IPv6 address.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the server or a fully qualified domain name to which you want to connect.

The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two.

Default

Empty string

Notes

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

GUI Tab

General tab

See Also

- Proxy Mode on page 234
- Proxy Password on page 235
- Proxy Port on page 236
- Proxy User on page 237

Proxy Mode

Supported on Windows, UNIX, and Linux only.

Attribute

ProxyMode (PXM)

Purpose

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

Valid Values

`0` | `1`

Behavior

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

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

Default

`0` (NONE)

GUI Tab

General tab

See Also

- Proxy Host on page 234
- Host on page 213
- Proxy Password on page 235
- Proxy Port on page 236
- Proxy User on page 237

Proxy Password

Supported on Windows, UNIX, and Linux only.

Attribute

ProxyPassword (PXPW)

Purpose

Specifies the password needed to connect to the proxy server.

Valid Values

`String`

where:

`String`

specifies the password to use to connect to the Proxy Server. Contact your system administrator to obtain your password.

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy Password option is ignored.
- Proxy Password is required only when the proxy server has been configured to require authentication.

Default

Empty string

GUI Tab

General tab

See Also

- Proxy Host on page 234
- Proxy Mode on page 234
- Proxy Port on page 236
- Proxy User on page 237

Proxy Port

Supported on Windows, UNIX, and Linux only.

Attribute

ProxyPort (PXPT)

Purpose

Specifies the port number where the proxy server is listening for HTTP requests.

Valid Values

`port_name`

where:

`port_name`

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

Notes

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

Default

`0`

GUI Tab

General tab

See Also

- Proxy Host on page 234
- Proxy Mode on page 234
- Proxy Password on page 235
- Proxy User on page 237

Proxy User

Supported on Windows, UNIX, and Linux only.

Attribute

ProxyUser (PXU)

Purpose

Specifies the user name needed to connect to the Proxy Server.

Valid Values

The default user ID that is used to connect to the Proxy Server.

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy User option is ignored.
- Proxy User is required only when the proxy server has been configured to require authentication.

Default

Empty string

GUI Tab

General tab

See Also

- Proxy Host on page 234

- Proxy Mode on page 234
- Proxy Password on page 235
- Proxy Port on page 236

PRNGSeedFile

UNIX®

Supported on UNIX and Linux only.

Attribute

PRNGSeedFile (PSF)

Purpose

Specifies the absolute path for the entropy-source file or device used as a seed for 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 239

PRNGSeedSource

UNIX®

Supported on UNIX and Linux only.

Attribute

PRNGSeedSource (PSS)

Purpose

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 238

Procedure Returns Results

Attribute

ProcedureRetResults (PRR)

Purpose

Determines whether the driver returns result sets from stored procedures/functions.

See "Support of materialized views" for details.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), the driver returns result sets from stored procedures/functions. When set to `1` and you execute a stored procedure that does not return result sets, you will incur a small performance penalty.

If set to `0` (Disabled), the driver does not return result sets from stored procedures.

Notes

- This connection option can affect performance.

Default

`0` (Disabled)

GUI Tab

Advanced tab

See also

Performance considerations on page 92

See also

Support of materialized views on page 105

Program ID

Attribute

ProgramID (PID)

Purpose

The product and version information of the driver on the client to be stored in the database. This value sets the PROCESS value in the V$SESSION table on the server. This value is used by the client information feature.

Valid Values

`string`

where:

`string`

is a value that identifies the product and version of the driver on the client.

If a value for this option is not specified, the driver uses the process ID of the session.

Notes

- This connection option can affect performance.

Default

None

GUI Tab

Client Monitoring tab

See also

Performance considerations on page 92

Query Timeout

Attribute

QueryTimeout (QT)

Description

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

where:

x

is a number of seconds.

Behavior

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

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

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

Default

`0`

GUI Tab

Advanced tab

Record Delimiter

Supported on Windows, UNIX, and Linux only.

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

Report Recycle Bin

Attribute

ReportRecycleBin (RRB)

Purpose

Determines whether support is provided for reporting objects that are in the Oracle Recycle Bin.

On Oracle 10*g* R1 and higher, when a table is dropped, it is not actually removed from the database, but placed in the recycle bin instead.

Valid Values

`0` | `1`

Behavior

If set to `1` (Enabled), support is provided for reporting objects that are in the Oracle Recycle Bin.

If set to `0` (Disabled), the driver does not return tables contained in the recycle bin in the result sets returned from SQLTables and SQLColumns. Functionally, this means that the driver filters out any results whose Table name begins with BIN$.

Default

`0` (Disabled)

GUI Tab

Advanced tab

SDU Size

Attribute

SDUSize (SDU)

Purpose

Specifies the size in bytes of the Session Data Unit (SDU) that the driver requests when connecting to the server. The SDU size is equivalent to the maximum number of bytes in a database protocol packets sent across the network. The setting of this option serves only as a suggestion to the database server. The actual SDU is negotiated with the database server.

Valid Values

x

where:

x

is an integer from `512` to `2097152` for Oracle 12*c* and higher, or, for earlier database versions, an integer from `512` to `32767`.

Behavior

When connecting to the server, the driver requests the specified value to be used as the maximum SDU size. While the specified value is only a suggestion, it affects the actual SDU size that is negotiated with the server.

To optimize performance, set this option based on the size of result sets returned by your application. If your application returns large result sets, set this option to the maximum SDU size configured on the database server. This reduces the total number of round trips required to return data to the client, thus improving performance. If your application returns small result sets, set this option to a size smaller than the maximum to avoid burdening your network with unnecessarily large packets.

Notes

- This option is mutually exclusive with the Server Name and TNSNames File connection option. The driver generates an exception if a value is specified for SDU Size in conjunction with either option.

Default

`16384`

GUI Tab

Performance tab

See also

Performance considerations on page 92

Server Name

Attribute

ServerName (SRVR)

Purpose

Specifies a net service name that exists in the TNSNAMES.ORA file. The corresponding net service name entry in the TNSNAMES.ORA file is used to obtain Host, Port Number, and Service Name or SID information.

Valid Values

server_name

where:

server_name

is a net service name in the TNSNAMES.ORA file.

Notes

- This option is mutually exclusive with the LDAP Distinguished Name, Host, Port Number, SID, and Service Name options.

Default

None

GUI Tab

General tab

Server Process Type

Attribute

ServerType (ST)

Purpose

Determines whether the connection is established using a shared or dedicated server process (dedicated thread on Windows).

Valid Values

0 | 1 | 2

Behavior

If set to 0 (Server Default), the driver uses the default server process set on the server.

If set to 1 (Shared), the server process used is retrieved from a pool. The socket connection between the application and server is made to a dispatcher process on the server. This setting allows there to be fewer processes than the number of connections, reducing the need for server resources. Use this value when a server must handle a large number of connections.

If set to 2 (Dedicated), a server process is created to service only that connection. When that connection ends, so does the process (UNIX and Linux) or thread (Windows). The socket connection is made directly between the application and the dedicated server process or thread. When connecting to UNIX and Linux servers, a dedicated server process can provide significant performance improvement, but uses more resources on the server. When connecting to Windows servers, the server resource penalty is insignificant. Use this value if you have a batch environment with a low number of connections.

Notes

- The server must be configured for shared connections (the SHARED_SERVERS initialization parameter on the server has a value greater than 0) for the driver to be able to specify the shared server process type.
- This connection option can affect performance.

Default

0 (Server Default)

GUI Tab

Advanced tab

See also

Performance considerations on page 92

Service Name

Attribute

ServiceName (SN)

Purpose

The Oracle service name that specifies the database used for the connection. The service name is a string that is the global database name—a name that is comprised of the database name and domain name, for example:

```
sales.us.acme.com
```

The service name is included as part of the Oracle connect descriptor, which is a description of the destination for a network connection. The service name is specified in the CONNECT_DATA parameter of the connect descriptor, for example:

```
(CONNECT_DATA=(SERVICE_NAME=sales.us.acme.com))
```

In this example, you would specify `sales.us.acme.com` as the value for the Service Name connection option.

Valid Values

service_name | %DEFAULT%

where:

service_name

is the description of the destination for a network connection.

Behavior

If set to *sid*, the driver attempts to connect to the Oracle instance that corresponds to the specified Oracle System Identifier.

If set to %DEFAULT%, the driver attempts to connect to the Service Name specified by the DEFAULT_SERVICE_LISTENER property in the server-side `listener.ora` file.

Notes

- This option is mutually exclusive with the LDAP Distinguished Name, SID, Server Name, and TNSNames File options.

Default

None. If no values are specified for the SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.

GUI Tab

General tab

SID

Attribute

SID (SID)

Purpose

The Oracle System Identifier that refers to the instance of Oracle running on the server.

Valid Values

sid | %DEFAULT%

where:

sid

is the name of the Oracle System Identifier.

Behavior

If set to *sid*, the driver attempts to connect to the Oracle instance that corresponds to the specified Oracle System Identifier.

If set to %DEFAULT%, the driver attempts to connect to the SID specified by the DEFAULT_SERVICE_LISTENER property in the server-side `listener.ora` file.

Notes

- This option is mutually exclusive with the LDAP Distinguished Name, Service Name, Server Name, and TNSNames File options.

Default

None. If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the `ORCL` SID by default.

GUI Tab

General 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 also

- CryptoLibName on page 192
- Advanced tab on page 53

Support Binary XML

Attribute

SupportBinaryXML (SBX)

Purpose

Enables the driver to support XMLType with binary storage on servers running Oracle 12*c* and higher.

Valid Values

`0 | 1`

Behavior

If set to `0` (Disabled), the driver does not support XMLType with binary storage and returns the error "This column type is not currently supported by this driver."

If set to 1 (Enabled), the driver supports XMLType with binary storage by negotiating server and client capabilities during connection time. As a result of this negotiation, decoded data associated with XMLType columns is returned in an in-line fashion without locators. This setting is supported only for Oracle 12c and higher.

Notes

- Queries involving XMLType with binary storage and XMLType with CLOB storage are affected when Support Binary XML is enabled (SupportBinaryXML=1). When Support Binary XML is enabled, XMLType with binary storage and XMLType with CLOB storage are returned in an in-line fashion without locators. Under these circumstances, executing a Select that includes XMLType columns can degrade performance because the driver must pull all in-line data to execute the query."

Default

0 (Disabled)

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

Timestamp Escape Mapping

Attribute

TimestampEscapeMapping (TEM)

Purpose

Determines how the driver maps Date, Time, and Timestamp literals.

Valid Values

`0` | `1`

Behavior

If set to `0` (Oracle Version Specific), the driver determines whether to use the TO_DATE or TO_TIMESTAMP function based on the version of the Oracle server to which it is connected. If the driver is connected to an 8.x server, it maps the Date, Time, and Timestamp literals to the TO_DATE function. If the driver is connected to a 9.x or higher server, it maps these escapes to the TO_TIMESTAMP function.

If set to `1` (Oracle 8x Compatible), the driver always uses the Oracle 8.x TO_DATE function as if connected to an Oracle 8.x server.

Default

`0` (Oracle Version Specific)

GUI Tab

Advanced tab

TNSNames File

Attribute

TNSNamesFile (TNF)

Purpose

Specifies the name of the `TNSNAMES.ORA` file. In a `TNSNAMES.ORA` file, connection information for Oracle services is associated with an Oracle net service name. The entry in the `TNSNAMES.ORA` file specifies Host, Port Number, and Service Name or SID.

TNSNames File is ignored if no value is specified in the Server Name option. If the Server Name option is specified but the TNSNames File option is left blank, the TNS_ADMIN environment setting is used for the `TNSNAMES.ORA` file path. If there is no TNS_ADMIN setting, the ORACLE_HOME environment setting is used. On Windows, if ORACLE_HOME is not set, the path is taken from the Oracle section of the Registry.

Using an Oracle `TNSNAMES.ORA` file to centralize connection information in your Oracle environment simplifies maintenance when changes occur. If, however, the `TNSNAMES.ORA` file is unavailable, then it is useful to be able to open a backup version of the `TNSNAMES.ORA` file (TNSNames file failover). You can specify one or more backup, or alternate, `TNSNAMES.ORA` files.

Valid Values

path_filename

where:

path_filename

is the entire path, including the file name, to the TNSNAMES.ORA file.

Behavior

To specify multiple TNSNAMES.ORA file locations, separate the names with a comma and enclose the locations in parentheses (you do not need parentheses for a single entry). For example:

```
(F:\server2\oracle\tnsnames.ora, C:\oracle\product\10.1\db_1\network\admin\tnsnames.ora)
```

The driver tries to open the first file in the list. If that file is not available, then it tries to open the second file in the list, and so on.

Connection Retry Count and Connection Retry Delay are also valid with TNSNames failover. The driver makes at least one attempt to open the files, and, if Connection Retry Count is enabled, more than one. If Connection Retry Delay is enabled, the driver waits the specified number of seconds between attempts. Load Balancing is not available for TNSNames failover.

Notes

- **Warning**: If you are distributing the driver with your application, you must prevent your end users from setting the value for the TNSNamesFile option. The TNSNamesFile option provides a method for you to specify a TNSNAMES.ORA file used for connecting to your Oracle service. 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 mutually exclusive with the LDAP Distinguished Name, Host, Port Number, SID, and Service Name (ServiceName) options.
- By default, if no value is specified for the Service name or SID in the tnsnames.ora file, the driver will use the SID or service name specified by the DEFAULT_SERVICE_LISTENER property in the server-side listener.ora file.

Default

None. If no values is specified for either the LDAP Distinguished Name, SID, Service Name, or TNSNames option, the driver attempts to connect to the ORCL SID by default.

GUI Tab

General tab

See Also

- Connection Retry Count on page 188
- Connection Retry Delay on page 189

Trust Store

Attribute

Truststore (TS)

Purpose

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

Valid Values

path|`data://-----BEGIN CERTIFICATE-----`*certificate_content*`-----END CERTIFICATE-----`

where:

`path`

is the absolute path to the truststore file.

`certificate_content`

is the content of the TLS/SSL certificate.

Notes

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

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

  Note that the number of dashes (`-----`) must be the same before and after both `BEGIN CERTIFICATE` and `END CERTIFICATE`.
- When specifying the certificate content for authentication, do not specify the truststore password. Since the truststore file is not required to be stored on the disk when the certificate content is specified directly, the driver need not unlock its contents.
- The Trust Store field on the Driver setup dialog supports content up to 8192 characters in length. For specifying certificate content longer than 8192 characters, edit the registry and manually add the entry to the DSN.
- On Windows platforms, if the required certificates are available in the Windows certificate store, the Trust Store and Truststore Password options need not be used.
- The value specified for this option should be an absolute path to a mounted drive.
- **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.

Default

None

GUI Tab

Security tab

Trust Store Password

TruststorePassword (TSP)

Purpose

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

Valid Values

`truststore_password`

where:

`truststore_password`

is a valid password for the truststore file.

Notes

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

Default

None

GUI Tab

Security tab

Use Current Schema for SQLProcedures

Attribute

UseCurrentSchema (UCS)

Description

Determines whether the driver returns only procedures owned by the current user when executing SQLProcedures.

Valid Values

`0 | 1`

Behavior

When set to `1` (Enabled), the call for SQLProcedures is optimized, but only procedures owned by the current user are returned.

When set to `0` (Disabled), the driver does not limit the procedures returned.

Default

`1` (Enabled)

GUI Tab

Performance tab

See also

Performance considerations on page 92

User Name

Attribute

LogonID (UID)

Description

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

Valid Values

`userid`

where:

`userid`

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

Default

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 TLS/SSL encryption is enabled (`EncryptionMethod=1`). When using TLS/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

Wallet Password

Attribute

CredentialsWalletPassword (CWPWD)

Purpose

Specifies the password used to access the Oracle Wallet in which your database credential information is stored. When `AuthenticationMethod=16` (Wallet UID & PWD), the driver uses this value to retrieve the database user ID and password that is stored in the wallet file specified by the Credentials Wallet Path (CredentialsWalletPath) option.

Valid Values

`credentials_password`

where:

`credentials_password`

is the password used to access the Oracle Wallet in which your database credential information is stored. This value is typically the password used to create your wallet.

Notes

- This option is required only if you are using an `ewallet.p12` file for your wallet. For wallets using a `cwallet.sso` file, the password for the wallet is stored in this file and, therefore, no value for this option needs to be provided.
- If your wallet contains multiple user ID and password pairs, specify the entry containing the correct credentials using the Credentials Wallet Entry (CredentialsWalletEntry) option.

Default

None

GUI Tab

Logon dialog

See Also

- Oracle Wallet Password Store on page 122

Wire Protocol Mode

Attribute

WireProtocolMode (WPM)

Description

Specifies whether the driver optimizes network traffic to the Oracle server.

Valid Values

`1` | `2`

Behavior

If set to `1`, the driver operates in normal wire protocol mode without optimizing network traffic.

If set to 2, the driver optimizes network traffic to the Oracle server for result sets that contain repeating data in some or all of the columns, and the repeating data is in consecutive rows. It also optimizes network traffic if the application is updating or inserting images, pictures, or long text or binary data.

Notes

- This connection option can affect performance.

Default

2

GUI Tab

Performance tab

See Also

Performance considerations on page 92