

Progress[®] DataDirect[®] for ODBC for Oracle[™] Wire Protocol Driver User's Guide and Reference

Release 8.0.2

Copyright

© 2020 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, Icenium, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. Analytics360, AppServer, BusinessEdge, DataDirect Autonomous REST Connector, DataDirect Spy, SupportLink, DevCraft, Fiddler, iMail, JustAssembly, JustDecompile, JustMock, NativeScript Sidekick, OpenAccess, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Updated: 2020/07/27

Table of Contents

Preface.....	13
Welcome to the Progress DataDirect for ODBC Oracle Wire Protocol Driver: Version 8.0.2.....	13
What's New in this Release?.....	14
Conventions Used in This Guide.....	18
About the Product Documentation.....	19
Contacting Technical Support.....	20
Getting Started.....	23
Configuring and Connecting on Windows.....	24
Configuring a Data Source.....	24
Testing the Connection.....	25
Configuring and Connecting on UNIX and Linux	25
Environment Configuration.....	25
Test Loading the Driver.....	26
Configuring a Data Source in the System Information File.....	26
Testing the Connection.....	27
Configuring and Connecting on macOS.....	28
iODBC Driver Manager	28
Configuring a Data Source	28
Testing the Connection	29
Accessing Data With Third-Party Applications.....	30
What Is ODBC?.....	31
How Does It Work?.....	32
Why Do Application Developers Need ODBC?.....	32
About the Oracle Wire Protocol Driver.....	33
Driver Requirements.....	34
ODBC Compliance.....	34
Support for Multiple Environments.....	35
Support for Windows Environments.....	35
Support for UNIX and Linux Environments.....	37
Support for macOS Environments.....	42
Version String Information.....	43
getFileVersionString Function.....	45
Data Types.....	45
XMLType.....	46

Retrieving Data Type Information.....	48
Isolation and Lock Levels Supported.....	49
Using Parameter Arrays.....	49
Supported Features.....	51
Unicode Support.....	51
Using IP Addresses.....	52
Number of Connections and Statements Supported.....	52
Support for Oracle RAC.....	52
SQL Support.....	53
MTS Support.....	53
OS Authentication.....	53
Stored Procedure Results.....	53
Support of Materialized Views.....	54
Using the Driver.....	55
Configuring and Connecting to Data Sources.....	56
Configuring the Product on UNIX/Linux.....	56
Configuring the Product on macOS.....	65
Data Source Configuration on Windows.....	75
Using a Connection String.....	113
Using a Logon Dialog Box.....	113
Performance Considerations.....	115
Using LDAP.....	119
Connecting through a proxy server.....	119
Oracle Connection Manager	121
Unexpected Characters.....	122
Using Failover.....	123
Connection Failover.....	124
Extended Connection Failover.....	125
Select Connection Failover.....	126
Guidelines for Primary and Alternate Servers.....	127
Using Client Load Balancing	127
Using Connection Retry.....	128
Configuring Failover-Related Options.....	128
Using Client Information.....	132
How Databases Store Client Information.....	132
Storing Client Information.....	132
Using Security.....	133
Authentication.....	133
Data Encryption Across the Network.....	137
Data Encryption and Integrity	137
Summary of Security-Related Options.....	141

Using DataDirect Connection Pooling.....	145
Creating a Connection Pool.....	146
Adding Connections to a Pool.....	146
Removing Connections from a Pool.....	146
Handling Dead Connections in a Pool.....	147
Connection Pool Statistics.....	148
Summary of Pooling-Related Options.....	148
Using DataDirect Bulk Load.....	149
Bulk Export and Load Methods.....	150
Exporting Data from a Database.....	151
Bulk Loading to a Database.....	152
The Bulk Load Configuration File	153
Sample Applications.....	156
Character Set Conversions.....	156
External Overflow Files.....	156
Limitations.....	157
Summary of Related Options for DataDirect Bulk Load.....	157
Using Bulk Load for Batch Inserts.....	158
Determining the Bulk Load Protocol.....	158
Limitations.....	159
Summary of Related Options for Bulk Load for Batch Inserts	159
Persisting a Result Set as an XML Data File.....	160
Using the Windows XML Persistence Demo Tool.....	160
Using the UNIX/Linux XML Persistence Demo Tool.....	162
Troubleshooting.....	163
Diagnostic Tools.....	163
ODBC Trace.....	163
The Test Loading Tool.....	167
ODBC Test.....	168
iODBC Demo and iODBC Test.....	168
The Example Application.....	168
Other Tools.....	168
Error Messages.....	169
Troubleshooting.....	170
Setup/Connection Issues.....	170
Interoperability Issues.....	172
Performance Issues.....	173
Connection Option Descriptions.....	175
Accounting Info.....	183
Action.....	184
AllowedOpenSSLVersions.....	184

Alternate Servers.....	186
Application Name.....	186
Application Using Threads.....	187
Array Size.....	188
Authentication Method.....	188
Batch Size.....	189
Bulk Binary Threshold.....	190
Bulk Character Threshold.....	191
Bulk Options.....	192
Cached Cursor Limit.....	193
Cached Description Limit.....	193
Catalog Functions Include Synonyms.....	194
Catalog Options.....	195
Client Host Name.....	195
Client ID.....	196
Client User.....	197
Connection Pooling.....	198
Connection Reset.....	198
Connection Retry Count.....	199
Connection Retry Delay.....	200
Credentials Wallet Entry.....	201
Credentials Wallet Path.....	202
Crypto Protocol Version.....	203
CryptoLibName.....	204
Data Integrity Level.....	205
Data Integrity Types.....	206
Data Source Name.....	207
Default Buffer Size for Long/LOB Columns (in Kb).....	207
Describe at Prepare.....	208
Description.....	208
Edition Name.....	209
Enable Bulk Load.....	210
Enable N-CHAR Support.....	211
Enable Scrollable Cursors.....	212
Enable Server Result Cache.....	212
Enable SQLDescribeParam.....	213
Enable Static Cursors for Long Data.....	214
Enable Timestamp with Timezone.....	214
Encryption Level.....	215
Encryption Method.....	216
Encryption Types.....	217
Failover Granularity.....	218
Failover Mode.....	219
Failover Preconnect.....	220
Fetch TSWTZ as Timestamp.....	220

Field Delimiter.....	221
GSS Client Library.....	222
Host	222
Host Name In Certificate.....	223
IANAAppCodePage.....	224
Impersonate User.....	225
Initialization String.....	226
Key Password.....	226
Key Store.....	227
Key Store Password.....	228
LDAP Distinguished Name.....	228
Load Balancing.....	229
LoadBalance Timeout.....	230
LOB Prefetch Size.....	231
Local Timezone Offset.....	231
Lock Timeout.....	232
Login Timeout.....	233
Max Pool Size.....	234
Min Pool Size.....	234
Module.....	235
Password.....	236
Port Number	237
Proxy Host.....	237
Proxy Mode.....	238
Proxy Password.....	239
Proxy Port.....	240
Proxy User.....	241
PRNGSeedFile.....	242
PRNGSeedSource	243
Procedure Returns Results.....	244
Program ID.....	245
Query Timeout.....	246
Record Delimiter.....	246
Report Codepage Conversion Errors.....	247
Report Recycle Bin.....	248
SDU Size.....	248
Server Name.....	249
Server Process Type.....	250
Service Name.....	251
SID.....	252
SSLibName.....	253
Support Binary XML.....	254
TCP Keep Alive.....	254
Timestamp Escape Mapping.....	255
TNSNames File.....	256

Trust Store.....	257
Trust Store Password.....	258
Use Current Schema for SQLProcedures.....	258
User Name.....	259
Validate Server Certificate.....	259
Wallet Password.....	260
Wire Protocol Mode.....	261

Part I: Reference.....263

Code Page Values.....267

IANAAppCodePage Values.....	267
-----------------------------	-----

ODBC API and Scalar Functions.....273

API Functions.....	273
Scalar Functions.....	276
String Functions.....	277
Numeric Functions.....	279
Date and Time Functions.....	280
System Functions.....	282

Internationalization, Localization, and Unicode.....283

Internationalization and Localization.....	283
Locale.....	284
Language.....	284
Country.....	284
Variant.....	285
Unicode Character Encoding.....	285
Background.....	285
Unicode Support in Databases.....	286
Unicode Support in ODBC.....	286
Unicode and Non-Unicode ODBC Drivers.....	287
Function Calls.....	287
Data.....	289
Default Unicode Mapping.....	290
Driver Manager and Unicode Encoding on UNIX/Linux.....	291
References.....	292
Character Encoding in the odbc.ini and odbcinst.ini Files.....	292

Designing ODBC Applications for Performance Optimization.....295

Using Catalog Functions.....	296
------------------------------	-----

Caching Information to Minimize the Use of Catalog Functions.....	296
Avoiding Search Patterns.....	297
Using a Dummy Query to Determine Table Characteristics.....	297
Retrieving Data.....	298
Retrieving Long Data.....	298
Reducing the Size of Data Retrieved.....	298
Using Bound Columns.....	299
Using SQLExtendedFetch Instead of SQLFetch.....	299
Choosing the Right Data Type.....	300
Selecting ODBC Functions.....	300
Using SQLPrepare/SQLExecute and SQLExecDirect.....	300
Using Arrays of Parameters.....	300
Using the Cursor Library.....	301
Managing Connections and Updates.....	302
Managing Connections.....	302
Managing Commits in Transactions.....	302
Choosing the Right Transaction Model.....	303
Using Positioned Updates and Deletes.....	303
Using SQLSpecialColumns.....	303
Using Indexes.....	263
Introduction.....	264
Improving Row Selection Performance.....	264
Indexing Multiple Fields.....	265
Deciding Which Indexes to Create.....	265
Improving Join Performance.....	266
Locking and Isolation Levels.....	305
Locking.....	305
Isolation Levels.....	306
Locking Modes and Levels.....	307
SSL Encryption Cipher Suites.....	309
DataDirect Bulk Load.....	317
DataDirect Bulk Load Functions.....	317
Utility Functions.....	318
GetBulkDiagRec and GetBulkDiagRecW.....	318
Export, Validate, and Load Functions.....	320
ExportTableToFile and ExportTableToFileW.....	320
ValidateTableFromFile and ValidateTableFromFileW	323
LoadTableFromFile and LoadTableFromFileW.....	325
DataDirect Bulk Load Statement Attributes.....	328
SQL_BULK_EXPORT.....	328

SQL_BULK_EXPORT_PARAMS.....	329
Threading.....	331
WorkAround Options.....	333
Index.....	337

Preface

For details, see the following topics:

- [Welcome to the Progress DataDirect for ODBC Oracle Wire Protocol Driver: Version 8.0.2](#)
- [What's New in this Release?](#)
- [Conventions Used in This Guide](#)
- [About the Product Documentation](#)
- [Contacting Technical Support](#)

Welcome to the Progress DataDirect for ODBC Oracle Wire Protocol Driver: Version 8.0.2

This is your user's guide and reference for the Progress® DataDirect® *for* ODBC Oracle™ Wire Protocol driver.

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

- [Getting Started](#) on page 23 explains the basics for quickly configuring and testing the drivers.
- [What Is ODBC?](#) on page 31 provides an explanation of ODBC.
- [About the Oracle Wire Protocol Driver](#) on page 33 explains the driver, supported environments and driver requirements.
- [Supported Features](#) on page 51 explains features supported by the driver.
- [Using the Driver](#) on page 55 guides you through configuring the driver. It also explains how to use the functionality supported by the driver such as Authentication and SSL encryption.

- [Troubleshooting](#) on page 163 explains the tools to solve common problems and documents error messages.
- The [Connection Option Descriptions](#) on page 175 section contains detailed descriptions of the connection options supported by the driver.
- The [Reference](#) on page 263 section includes reference information about APIs, code page values, and performance tuning.

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

For the latest information about your driver, refer to the readme file in your software package.

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

What's New in this Release?

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

- [Release Notes](#) (includes the latest OpenSSL support information)
- [Supported Configurations](#)
- [DataDirect Support Matrices](#)

Changes since 8.0.2 GA

- **Driver Enhancements**
 - 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 136 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 119.
 - 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. See [Driver Manager and Unicode Encoding on UNIX/Linux](#) on page 291 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 121 for details.
 - The new AllowedOpenSSLVersions option allows you to determine which version of the OpenSSL library file the driver uses for data encryption. See [AllowedOpenSSLVersions](#) on page 184 or [Designating an OpenSSL Library](#) on page 139 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 [Troubleshooting](#) on page 163:
 - `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 210 and [IANAAppCodePage](#) on page 224 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 237, [Proxy Mode](#) on page 238, [Proxy Password](#) on page 239, [Proxy Port](#) on page 240, and [Proxy User](#) on page 241 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 225 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 251, [SID](#) on page 252, and [TNSNames File](#) on page 256 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**
 - On the GUI, proxy-related options have been moved from the General tab to the new Proxy tab.
 - The following Windows platforms have reached the end of their product lifecycle and are no longer supported by the driver:
 - Windows 8.0 (versions 8.1 and higher are still supported)
 - Windows Vista (all versions)
 - Windows XP (all versions)
 - Windows Server 2003 (all versions)
 - 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 188 for details.

Changes for 8.0.2 GA

- **Certifications**
 - Certified with Oracle 12c R2 (12.2)
 - Certified with Windows Server 2016
 - Certified with Red Hat Enterprise 7.3
 - Support for the following platforms is now generally available:
 - HP-UX IPF (32 and 64-bit)
 - HP-UX PA-RISC (32-bit)
 - Oracle Solaris x86 (32- and 64-bit)
 - Oracle Solaris on SPARC (32- and 64-bit)

- Support for the Intel Itanium II (IPF) processor on the following 64-bit Linux platforms is now generally available:
 - CentOS Linux 4.x, 5.x, 6.x, and 7.x
 - Oracle Linux 4.x, 5.x, 6.x, and 7.x
 - Red Hat Enterprise Linux AS, ES, and WS version 4.x, 5.x, 6.x, and 7.x

See [64-Bit Drivers Requirements for UNIX/Linux](#) on page 39 for details.

Support for operating environments and database versions are always being added. For the latest information about supported platforms and databases, refer to the Progress DataDirect database support matrices page at: <https://www.progress.com/matrices/datadirect>.

- **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 135 and [Using Oracle Wallet as a Keystore](#) on page 140 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 135 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 206 and [Data Integrity Level](#) on page 205 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 206 for details.

Changes for 8.0.1 GA

- **Certifications**

- Certified with Debian Linux 7.11, 8.5
- Certified with Ubuntu Linux 14.04, 16.04
- macOS v10.12.x (Sierra)

- **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 248 for details.
 - The new Support Binary XML connection option enables the driver to support XMLType with binary storage on servers running Oracle 12c and higher. See [Support Binary XML](#) on page 254 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 231 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 211 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 214 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 205 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 215 for details.

Changes for 8.0.0 GA

- **New Progress DataDirect for Oracle Wire Protocol Driver for Mac OS X**
 - The Oracle Wire Protocol driver is available for Mac OS X platforms, which includes:
 - Support for the following Mac OS X Platforms:
 - Mac OS X v10.11.x (El Capitan)
 - Mac OS X v10.10.x (Yosemite)
 - Mac OS X v10.9.x (Mavericks)
 - Support for the following databases:
 - Oracle 12c R1 (12.1)
 - Oracle 11g R1, R2 (11.1, 11.2)
 - Oracle 10g R1, R2 (10.1, 10.2)
 - Oracle 9i R1, R2 (9.0.1, 9.2)

- Oracle8i R3 (8.1.7)
- Support for iODBC Driver Manager, version 3.52.7 and higher.
- DataDirect Wire Protocol technology for improved response time and throughput.
- Support for core SQL-92 grammar.
- Supports ODBC Core and Level 1 functions.
- Advanced security features, including data encryption, Kerberos authentication, and Oracle Advanced Security.
- Support for failover protection.

See [Support for macOS Environments](#) on page 42 and [Configuring the Product on macOS](#) on page 65 for details.

Conventions Used in This Guide

The following sections describe the conventions used to highlight information that applies to specific operating systems and typographical conventions.

Operating System Symbols

The drivers are supported in the Windows, UNIX, Linux, and macOS environments. When the information provided is not applicable to all supported environments, the following symbols are used to identify that information:



The Windows symbol signifies text that is applicable only to Windows.



The UNIX symbol signifies text that is applicable only to UNIX and Linux.



The macOS symbol signifies text that is applicable only to macOS.

Typography

This guide uses the following typographical conventions:

Convention	Explanation
<i>italics</i>	Introduces new terms with which you may not be familiar, and is used occasionally for emphasis.
bold	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click Next .
BOLD UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
UPPERCASE	Indicates SQL reserved words.

Convention	Explanation
monospace	Indicates syntax examples, values that you specify, or results that you receive.
<i>monospaced italics</i>	Indicates names that are placeholders for values that you specify. For example, <i>filename</i> .
>	Separates menus and their associated commands. For example, Select File > Copy means that you should select Copy from the File menu.
/	The slash also separates directory levels when specifying locations under UNIX, Linux or macOS.
vertical rule	Indicates an "OR" separator used to delineate items.
brackets []	Indicates optional items. For example, in the following statement: <code>SELECT [DISTINCT], DISTINCT</code> is an optional keyword. Also indicates sections of the Windows Registry.
braces { }	Indicates that you must select one item. For example, { <code>yes no</code> } means that you must specify either <code>yes</code> or <code>no</code> .
ellipsis ...	Indicates that the immediately preceding item can be repeated any number of times in succession. An ellipsis following a closing bracket indicates that all information in that unit can be repeated.

About the Product Documentation

This guide provides specific information about your Progress DataDirect *for* ODBC driver. You can view this documentation in the HTML format installed with the product. The documentation is also available in PDF format . You can download the PDF version of the documentation at:

<https://www.progress.com/documentation/datadirect-connectors>

This guide is installed with the product as an HTML-based help system. This help system is located in the help subdirectory of the product installation directory. You can use the help system with any of the following browsers:

- Microsoft Edge on Windows 10
- Internet Explorer 7.x and higher
- Firefox 3.x and higher
- Safari 5.x
- Google Chrome 44.x and earlier

On all platforms, you can access the entire Help system by opening the following file from within your browser:

`install_dir/help/OracleHelp/index.html`

where *install_dir* is the path to the product installation directory.

Or, from a command-line environment, at a command prompt, enter:

```
browser_exe install_dir/help/OracleHelp/index.html
```

where *browser_exe* is the name of your browser executable and *install_dir* is the path to the product installation directory.

After the browser opens, the left pane displays the Table of Contents, Index, and Search tabs for the entire documentation library. When you have opened the main screen of the Help system in your browser, you can bookmark it in the browser for quick access later.

Note: Security features set in your browser can prevent the Help system from launching. A security warning message is displayed. Often, the warning message provides instructions for unblocking the Help system for the current session. To allow the Help system to launch without encountering a security warning message, the security settings in your browser can be modified. Check with your system administrator before disabling any security features.

Help is also available from the setup dialog box for each driver. When you click **Help**, your browser opens to the correct topic without opening the help Table of Contents. A grey toolbar appears at the top of the browser window.



This tool bar contains previous and next navigation buttons. If, after viewing the help topic, you want to see the entire library, click:



on the left side of the toolbar, which opens the left pane and displays the Table of Contents, Index, and Search tabs.

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

Getting Started

This chapter provides basic information about configuring your driver immediately after installation and testing your connection. To take full advantage of the features of the driver, read "About the Driver" and "Using the Driver".

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 and macOS, user and system data sources are stored on 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.

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

For details, see the following topics:

- [Configuring and Connecting on Windows](#)
- [Configuring and Connecting on UNIX and Linux](#)
- [Configuring and Connecting on macOS](#)
- [Accessing Data With Third-Party Applications](#)

Configuring and Connecting on Windows



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

Configuring a Data Source

To configure a data source:

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

- **User DSN:** If you installed 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 the appropriate driver and click **Finish** to display the driver Setup dialog box.

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

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

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

2. On the General tab, provide the following information; then, click **Apply**.

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.

Edition Name: Oracle 11g R2 and higher only. Type the name of the Oracle edition that the driver is to use when establishing a connection. Oracle 11g 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 tells the driver which edition of the schema objects to use.

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.

Testing the Connection

To test the connection:

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

See also

[Using a Logon Dialog Box](#) on page 113

[Connection Option Descriptions](#) on page 175

Configuring and Connecting on UNIX and Linux

UNIX[®]

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

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

See also

[Configuring the Product on UNIX/Linux](#) on page 56

Environment Configuration

To configure the environment:

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

```
echo $SHELL
```

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

```
env
```

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

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

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

From the C shell, you would enter:

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

Test Loading the Driver

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

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

```
ivtestlib /opt/odbc/lib/ivoraxx.so
```

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

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

Configuring a Data Source in the System Information File

The default `odbc.ini` file installed in the installation directory is a template in which you create data source definitions. You enter your site-specific database connection information using a text editor. Each data source definition must include the keyword `Driver=`, which is the full path to the driver.

The following examples show the minimum connection string options that must be set to complete a test connection, where *xx* represents *iv* for 32-bit or *dd* for 64-bit drivers, *yy* represents the driver level number, and *zz* represents the extension. The values for the options are samples and are not necessarily the ones you would use.

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

```
[Oracle]
Driver=ODBCHOME/lib/xxorayy.zz
EditionName=oracle 1
HostName=199.226.224.34
PortNumber=1521
ServiceName=sales.us.acme.com
```

Connection Option Descriptions:

EditionName: *Oracle 11g R2 and higher only.* The name of the Oracle edition the driver uses when establishing a connection. Oracle 11g 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 11g R2 and higher databases and tells the driver which edition of the schema objects to use.

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

PortNumber: The port number of your Oracle listener. Check with your database administrator for the number.

ServiceName: 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`.

SID: The Oracle System Identifier that refers to the instance of Oracle running on the server.

Note: SID and ServiceName are mutually exclusive. Only one or the other can be specified in the data source; otherwise, an error is generated.

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.

Testing the Connection

The driver installation includes an ODBC application called `example` that can be used to connect to a data source and execute SQL. The application is located in the `installation_directory/samples/example` directory.

To run the program after setting up a data source in the `odbc.ini`, 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`. If `example` is unable to connect, the appropriate error message is returned.

Configuring and Connecting on macOS



The following basic information enables you to configure a data source and test connect with a driver immediately after installation. On macOS, you can configure and modify data sources by editing the `odbc.ini` file or by configuring the GUI provided by iODBC Administrator. Connection values specified through the options on the of the Setup dialog box are stored either as a user or system data source in the `odbc.ini` file, or as a file data source in a specified location. See "Configuring and Connecting to Data Sources" for detailed information about configuring the macOS environment and data sources.

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

See also

[Configuring and Connecting to Data Sources](#) on page 56

iODBC Driver Manager

Before you can use the driver, you must install and setup the iODBC Driver Manager on your machine. iODBC is an open-source interface that manages data sources and loads DataDirect drivers for macOS applications. It is the most commonly used Driver Manager for macOS platforms and is included with some versions of the operating system. For more information, refer to <http://www.iodbc.org/>.

Configuring a Data Source

To configure a data source using the iODBC GUI:

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

The data source Setup dialog box appears. If you are configuring an existing data source, the dialog is prepopulated with a list of the connection option attribute-value pairs currently stored in the `odbc.ini` file. For new data sources, the dialog will be empty.


2. On the Setup dialog, provide keyword-value pairs for the connection options in the following table; then, click **OK**. To add missing pairs, click the Add button . Then, in the new row, type the attribute name in the Keyword field and the desired valid value in the Value field.

Table 1: Required Connection Options

Keyword	Value
EditionName	<i>Oracle 11g R2 and higher only.</i> The name of the Oracle edition the driver uses when establishing a connection. Oracle 11g 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 11g R2 and higher databases and tells the driver which edition of the schema objects to use.
HostName	The name or the IP address of the server to which you want to connect.
PortNumber	The port number of your Oracle listener. Check with your database administrator for the number.
ServiceName	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: <code>sales.us.acme.com</code> .
SID	The Oracle System Identifier that refers to the instance of Oracle running on the server. The default is <code>ORCL</code> .

Note: SID and ServiceName are mutually exclusive. Only one or the other can be specified in the data source; otherwise, an error is generated.

Testing the Connection

Note: Some earlier versions of the iODBC Administrator do not support testing 64-bit drivers. If you experience an issue, you can still test the connection by using the example application installed in the product installation directory. See "The example Application" for more information.

To test the connection:

1. After you have configured the data source, on the iODBC Data Source Administrator dialog, highlight your data source from the list; then, click **Test**. A logon dialog box appears.
2. Supply the requested information in the logon dialog box and click **OK**. Note that the information you enter in the logon dialog box during a test connect is not saved.
 - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.

- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.
3. On the driver iODBC Data Source Administrator, click **OK**. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the previously described procedure to modify your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

See also

[The example Application](#) on page 72

Accessing Data With Third-Party Applications

For procedures related to accessing data with common third-party applications, such as Tableau and Excel, refer to the Quick Start that corresponds to your data source and platform at <https://www.progress.com/documentation/datadirect-connectors>.

What Is ODBC?

The Open Database Connectivity (ODBC) interface by Microsoft allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data. ODBC permits maximum interoperability, which means a single application can access different DBMS. Application end users can then add ODBC database drivers to link the application to their choice of DBMS.

The ODBC interface defines:

- A library of ODBC function calls of two types:
 - Extended functions that support additional functionality, including scrollable cursors
 - Core functions that are based on the X/Open and SQL Access Group Call Level Interface specification
- SQL syntax based on the X/Open and SQL Access Group SQL CAE specification (1992)
- A standard set of error codes
- A standard way to connect and logon to a DBMS
- A standard representation for data types

The ODBC solution for accessing data led to ODBC database drivers, which are dynamic-link libraries on Windows and shared objects on UNIX and Linux, and dynamic libraries on macOS. These drivers allow an application to gain access to one or more data sources. ODBC provides a standard interface to allow application developers and vendors of database drivers to exchange data between applications and data sources.

For details, see the following topics:

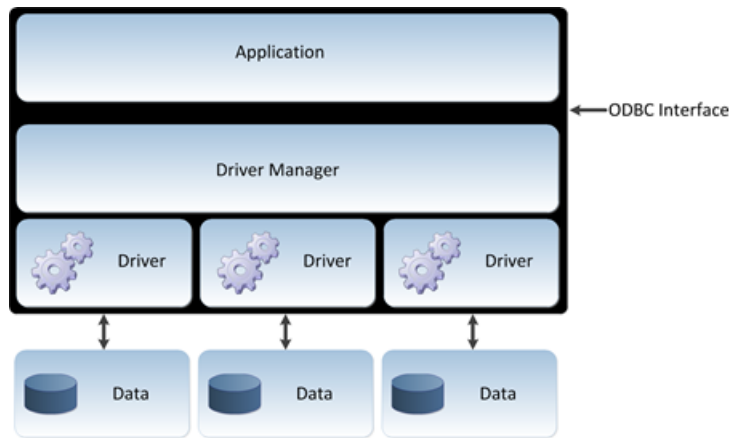
- [How Does It Work?](#)
- [Why Do Application Developers Need ODBC?](#)

How Does It Work?

The ODBC architecture has four components:

- An application, which processes and calls ODBC functions to submit SQL statements and retrieve results
- A Driver Manager, which loads drivers for the application
- A driver, which processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application
- A data source, which consists of the data to access and its associated operating system, DBMS, and network platform (if any) used to access the DBMS

The following figure shows the relationship among the four components:



Why Do Application Developers Need ODBC?

Using ODBC, you, as an application developer can develop, compile, and ship an application without targeting a specific DBMS. In this scenario, you do not need to use embedded SQL; therefore, you do not need to recompile the application for each new environment.

About the Oracle Wire Protocol Driver

The 32- and 64-bit Progress DataDirect *for* ODBC for Oracle Wire Protocol drivers (the Oracle Wire Protocol driver) support the following Oracle database servers:

- Oracle 12c R1 (12.1) and higher
- Oracle 11g R1 (11.1) and higher
- Oracle 10g R1 (10.1) and higher
- Oracle 9i R1 (9.0.1) and higher
- Oracle 8i R3 (8.1.7) and higher

The Oracle Wire Protocol driver is supported in the Windows, UNIX, Linux, and macOS environments. See "Support for Multiple Environments" for detailed information about the environments supported by this driver.

For the latest information on certifications and supported configurations, refer to the [Supported Matrices](#) and [Supported Configurations](#) pages.

See "Driver File Names for Windows" and "Driver File Names for UNIX/Linux" for the file name of the driver.

Note: The Oracle Wire Protocol driver does not require any Oracle client software. Progress DataDirect also provides an Oracle client-based driver. Refer to the [DataDirect Connect Series for ODBC Library](#) for details.

For details, see the following topics:

- [Driver Requirements](#)
- [ODBC Compliance](#)
- [Support for Multiple Environments](#)

- [Version String Information](#)
- [Data Types](#)
- [Isolation and Lock Levels Supported](#)
- [Using Parameter Arrays](#)

Driver Requirements

The driver has no client requirements.

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



MacOS

For macOS, the following functions are not supported by the iODBC driver manager, and therefore, cannot be successfully executed by the driver:

Core:

- SQLAllocConnect
- SQLAllocEnv
- SQLAllocStmt
- SQLFreeConnect
- SQLFreeEnv

Level 1:

- SQLGetStmtOption
- SQLSetStmtOption

Level 2:

- SQLDescribeParam
- SQLParamOptions

Level 3:

- SQLTransact

See "API Functions" for a list of the API functions supported by the driver.

See also

[API Functions](#) on page 273

Support for Multiple Environments

Your Progress DataDirect driver is ODBC-compliant for Windows, UNIX, Linux, and macOS operating systems. This section explains the environment-specific differences when using the database drivers in your operating environment.

Note: Support for operating environments and database versions are continually being added. For the latest information about supported platforms and databases, refer to the Progress DataDirect database support configurations Web pages at: <https://www.progress.com/matrices/supported-configurations>

See also

[Threading](#) on page 331

Support for Windows Environments

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

32-Bit Driver Requirements for Windows

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit driver. If your application was built with 64-bit system libraries, you must use the 64-bit driver (see "64-bit Driver Requirements"). The database to which you are connecting can be either 32-bit or 64-bit enabled.
- The following processors are supported:
 - x86: Intel
 - x64: Intel and AMD

- The following operating systems are supported for your DataDirect Connect *for* ODBC driver. All editions are supported unless otherwise noted.
 - Windows Server 2016
 - Windows Server 2012
 - Windows Server 2008
 - Windows 10
 - Windows 8.1
 - Windows 7
- An application that is compatible with components that were built using Microsoft Visual Studio 2015 compiler and the standard Win32 threading model.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

See also

[64-Bit Driver Requirements for Windows](#) on page 36

64-Bit Driver Requirements for Windows

- All required network software that is supplied by your database system vendors must be 64-bit compliant.
- The following processors are supported:
 - Intel
 - AMD
- The following operating systems are supported for your 64-bit driver. All editions are supported unless otherwise noted.
 - Windows Server 2016
 - Windows Server 2012
 - Windows Server 2008
 - Windows 10
 - Windows 8.1
 - Windows 7
- An application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 14.00.40310.41 and the standard Windows 64 threading model.
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Setup of the Driver

The driver must be configured before it can be used. See "Configuring a Data Source" for information about using the Windows ODBC Administrator. See "Configuring and Connecting to Data Sources" for details about driver configuration.

See also

[Configuring a Data Source](#) on page 24

[Configuring and Connecting to Data Sources](#) on page 56

Driver File Names for Windows

The prefix for all 32-bit driver file names is `iv`. The prefix for all 64-bit driver file names is `dd`. The file extension is `.dll`, which indicates dynamic link libraries. For example, the 32-bit Oracle Wire Protocol driver file name is `ivorann.dll`, where `nn` is the revision number of the driver.

For the 8.0 version of the 32-bit driver, the file name is:

```
ivora28.dll
```

For the 8.0 version of the 64-bit driver, the file name is:

```
ddora28.dll
```

Refer to the installed readme file for a complete list of installed files.

Support for UNIX and Linux Environments

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

32-Bit Driver Requirements for UNIX/Linux

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit drivers. If your application was built with 64-bit system libraries, you must use 64-bit drivers (see "64-bit Driver Requirements for UNIX/Linux"). The database to which you are connecting can be either 32-bit or 64-bit enabled.

AIX

- IBM POWER processor
- AIX 5L operating system, version 5.3 fixpack 5 and higher, 6.1, and 7.1
- An application compatible with components that were built using Visual Age C++ 6.0.0.0 and the AIX native threading model

HP-UX

- The following processors are supported:
 - PA-RISC
 - Intel Itanium II (IPF)

- The following operating systems are supported:
 - For PA-RISC: HP-UX 11i Versions 2 and 3 (B.11.23 and B.11.3x)
 - For IPF: HP-UX IPF 11i Versions 2 and 3 (B.11.23 and B.11.3x)
- For PA-RISC: An application compatible with components that were built using HP aC++ 3.60 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads).

All of the standard 32-bit UNIX drivers are supported on HP PA-RISC.
- For IPF: An application compatible with components that were built using HP aC++ 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads)

Linux

- The following processors are supported:
 - x86: Intel
 - x64: Intel and AMD
- The following operating systems are supported:
 - CentOS Linux 4.x, 5.x, 6.x, and 7.x
 - Debian Linux 7.11, 8.5
 - Oracle Linux 4.x, 5.x, 6.x, and 7.x
 - Red Hat Enterprise Linux 4.x, 5.x, 6.x, and 7.x
 - SUSE Linux Enterprise Server 10.x, and 11.x
 - Ubuntu Linux 14.04, 16.04
- An application compatible with components that were built using g++ GNU project C++ Compiler version 3.4.6 and the Linux native pthread threading model (Linuxthreads).

Oracle Solaris

- The following processors are supported:
 - Oracle SPARC
 - x86: Intel
 - x64: Intel and AMD
- The following operating systems are supported:
 - For Oracle SPARC: Oracle Solaris 8, 9, 10, 11.x
 - For x86/x64: Oracle Solaris 10, Oracle Solaris 11.x
- For Oracle SPARC: An application compatible with components that were built using Sun Studio 11, C++ compiler version 5.8 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

See also

[Threading](#) on page 331

[64-Bit Drivers Requirements for UNIX/Linux](#) on page 39

64-Bit Drivers Requirements for UNIX/Linux

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

AIX

- IBM POWER Processor
- AIX 5L operating system, version version 5.3 fixpack 5 and higher, 6.1, and 7.1
- An application compatible with components that were built using Visual Age C++ version 6.0.0.0 and the AIX native threading model

HP-UX

- HP-UX IPF 11i operating system, Versions 2 and 3 (B.11.23 and B.11.31)
- HP aC++ v. 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads)

Linux

- Intel Itanium II (IPF)
- Intel and AMD processors
- The following operating systems are supported:
 - For Intel Itanium II (IPF):
 - CentOS Linux 4.x, 5.x, 6.x, and 7.x
 - Oracle Linux 4.x, 5.x, 6.x, and 7.x
 - Red Hat Enterprise Linux AS, ES, and WS version 4.x, 5.x, 6.x, and 7.x
 - For x64:
 - CentOS Linux 4.x, 5.x, 6.x, and 7.x
 - Debian Linux 7.11 and 8.5
 - Oracle Linux 4.x, 5.x, 6.x, and 7.x
 - Red Hat Enterprise Linux AS, ES, and WS version 4.x, 5.x, 6.x, and 7.x
 - SUSE Linux Enterprise Server 10.x, and 11.x
 - Ubuntu Linux 14.04 and 16.04
- 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)

Oracle Solaris

- The following processors are supported:
 - Oracle SPARC
 - x64: Intel and AMD
- The following operating systems are supported:
 - For Oracle SPARC: Oracle Solaris 8, 9, 10, and 11.x
 - For x64: Oracle Solaris 10 and Oracle Solaris 11.x Express
- For Oracle SPARC: An application compatible with components that were built using Sun Studio 11, C++ compiler version 5.8 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

See also

[Threading](#) on page 331

AIX

If you are building 64-bit binaries, you must pass the define ODBC64. The example Application provides a demonstration of this. See the installed file `example.txt` for details.

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

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

HP-UX 11 aCC

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

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

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

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

Linux

If you are building 64-bit binaries, you must pass the define ODBC64. The example Application provides a demonstration of this. See the installed file `example.txt` for details.

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

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

Oracle Solaris

If you are building 64-bit binaries, you must pass the define ODBC64. The example Application provides a demonstration of this. See the installed file `example.txt` for details.

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

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

Setup of the Environment and the Drivers

On UNIX and Linux, several environment variables and the system information file must be configured before the drivers can be used. See the following topics for additional information:

- "Configuring and Connecting on UNIX and Linux" contains a brief description of these variables.
- "Configuring and Connecting to Data Sources" provides details about driver configuration.
- "Configuring the Product on UNIX/Linux" provides complete information about using the drivers on UNIX and Linux.

See also

[Configuring and Connecting on UNIX and Linux](#) on page 25

[Configuring the Product on UNIX/Linux](#) on page 56

[Configuring and Connecting to Data Sources](#) on page 56

Driver File Names for UNIX/Linux

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

For the 8.0 version of the 32-bit driver, the file name is:

```
ivora28.so
```

For the 8.0 version of the 64-bit driver, the file name is:

```
ddora28.so
```

Refer to the readme file for a complete list of installed files.

Support for macOS Environments



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

32-Bit Driver Requirements for macOS

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use the 32-bit driver. If your application was built with 64-bit system libraries, you must use the 64-bit driver (see "64-bit Driver Requirements"). The database to which you are connecting can be either 32-bit or 64-bit enabled.
- iODBC Driver Manager, version 3.52.7 or higher, is required.
- The following processors are supported:
 - Intel
- The following operating systems are supported for your 32-bit driver. All editions are supported unless otherwise noted.
 - macOS v10.12.x (Sierra)
 - Mac OS X v10.11.x (El Capitan)
 - Mac OS X v10.10.x (Yosemite)
 - Mac OS X v10.9.x (Mavericks)

64-bit Driver Requirements for macOS

- All required network software that is supplied by your database system vendors must be 64-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit driver. If your application was built with 64-bit system libraries, you must use the 64-bit driver (see "32-bit Driver Requirements for macOS"). The database to which you are connecting can be either 32-bit or 64-bit enabled.
- iODBC Driver Manager, version 3.52.7 or higher, is required.
- The following processors are supported:
 - Intel
- The following operating systems are supported for your 64-bit driver. All editions are supported unless otherwise noted.
 - macOS v10.12.x (Sierra)
 - Mac OS X v10.11.x (El Capitan)
 - Mac OS X v10.10.x (Yosemite)

- Mac OS X v10.9.x (Mavericks)

Setup of the Environment and the Driver

On macOS platforms, several environment variables and the system information file must be configured before the driver can be used. See the following topics for additional information:

- [Configuring and Connecting on macOS](#) on page 28 contains a brief description of these variables.
- [Data Source Configuration on macOS](#) on page 66 provides details about driver configuration.
- [Configuring the Product on macOS](#) on page 65 provides complete information about using the driver on macOS.

Driver Names for macOS

The drivers are ODBC API-compliant dynamic libraries. The prefix for file names is `dd`. The prefix for all 32-bit driver file names is `iv`. The prefix for all 64-bit driver file names is `dd`. The driver file names are lowercase and the extension is `.dylib`, the standard form for a dynamic library. For example, the 32-bit driver file name is `ivorann.dylib`, where `nn` is the revision number of the driver.

For the 8.0 version of the 32-bit driver, the file name is:

```
ivora28.dylib
```

For the 8.0 version of the 64-bit driver, the file name is:

```
ddora28.dylib
```

Refer to the installed readme file for a complete list of installed files.

Version String Information

The driver has a version string of the format:

```
XX.YY.ZZZZ(BAAAA, UBBBB)
```

or

```
XX.YY.ZZZZ (bAAAA, uBBBB)
```

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

```
XX.YY.ZZZZ(UBBBB)
```

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.



On UNIX, Linux and macOS, 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 `qesqltext.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 2: Oracle Data Types

Oracle	ODBC
BFILE ¹	SQL_LONGVARBINARY
BINARY DOUBLE ²	SQL_REAL
BINARY FLOAT ²	SQL_DOUBLE
BLOB ^{2, 3}	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB ^{2, 3}	SQL_LONGVARCHAR
DATE	SQL_TYPE_TIMESTAMP
LONG	SQL_LONGVARCHAR
LONG RAW	SQL_LONGVARBINARY
NCHAR	SQL_WVARCHAR
NCLOB	SQL_WLONGVARCHAR
NVARCHAR2	SQL_WVARCHAR
NUMBER	SQL_DOUBLE

¹ Read-Only

² Supported only on Oracle 10g and higher.

³ Supported in basic file and SecureFiles storage.

Oracle	ODBC
NUMBER (p,s)	SQL_DECIMAL
RAW	SQL_VARBINARY
TIMESTAMP ⁴	SQL_TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE ^{4, 5}	SQL_TIMESTAMP
TIMESTAMP WITH TIMEZONE ^{4,5}	SQL_VARCHAR
VARCHAR2	SQL_VARCHAR
XMLType ⁶	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 48

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.

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.

⁴ Supported only on Oracle 9i and higher.

⁵ Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option only on Oracle 10g R2 and higher.

⁶ XMLType columns with object relational storage are not supported.

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_ts7
  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>
  </LineItems>
</PurchaseOrder>
');
```

⁷ Note that the table space must be created before executing a statement similar to the one used in the example.

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.

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 ("%s %10i %10u\n", typeName, sqlDataType, columnSize);
            }

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


See also[ODBC Test](#) on page 168[Data Types](#) on page 45

Isolation and Lock Levels Supported

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

See also[Locking and Isolation Levels](#) on page 305

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. See "Using Arrays of Parameters" for more information about using arrays of parameters to 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.

See also[Using Arrays of Parameters](#) on page 300[Using Bulk Load for Batch Inserts](#) on page 158

Supported Features

This section describes some of the supported features that allow you to take full advantage of the driver.

For details, see the following topics:

- [Unicode Support](#)
- [Using IP Addresses](#)
- [Number of Connections and Statements Supported](#)
- [Support for Oracle RAC](#)
- [SQL Support](#)
- [MTS Support](#)
- [OS Authentication](#)
- [Stored Procedure Results](#)
- [Support of Materialized Views](#)

Unicode Support

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

See also

[Data Types](#) on page 45

Using IP Addresses

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

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

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

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol Driver;  
Host=2001:DB8:0000:0000:8:800:200C:417A;PORT=5439;  
DB=OracleACCT;UID=JOHN;PWD=XYZZYYou;SERVICENAME=SALES.US.ACME.COM
```

In addition to the normal IPv6 format, the drivers in the preceding tables support IPv6 alternative formats for compressed and IPv4/IPv6 combination addresses. For example, the following connection string specifies the server using IPv6 format, but uses the compressed syntax for strings of zero bits:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol Driver;  
Host=2001:DB8:0:0:8:800:200C:417A;PORT=5439;  
DB=OracleACCT;UID=JOHN;PWD=XYZZYYou;SERVICENAME=SALES.US.ACME.COM
```

Similarly, the following connection string specifies the server using a combination of IPv4 and IPv6:

```
DRIVER=DataDirect 8.0 Oracle Wire Protocol Driver;  
Host=2001:DB8:0:0:8:800:123.123.78.90;PORT=5439;  
DB=OracleACCT;UID=JOHN;PWD=XYZZYYou;SERVICENAME=SALES.US.ACME.COM
```

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

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

Number of Connections and Statements Supported

The Oracle Wire Protocol driver supports multiple connections and multiple statements per connection.

Support for Oracle RAC

Oracle introduced Real Application Clusters (RAC) with Oracle 9i, 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 123

SQL Support

The driver supports the core SQL grammar.

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.

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.

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.

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](#)
- [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](#)

Configuring and Connecting to Data Sources

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

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

See also

[Getting Started](#) on page 23

[Using a Connection String](#) on page 113

[Connection Option Descriptions](#) on page 175

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 56

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 58

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 63

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 58

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 all UNIX/Linux drivers use if individual data sources have not specified a different value. See "IANAAppCodePage" and "Code Page Values" for details. The default value is 4.

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 description of these keywords and discussion of tracing, see "ODBC Trace".

See also

[Connection Option Descriptions](#) on page 175

[Sample `odbcinst.ini` File](#) on page 63

[File Data Sources](#) on page 64

[IANAAppCodePage](#) on page 224

[Code Page Values](#) on page 267

[DSN-less Connections](#) on page 63

[ODBC Trace](#) on page 163

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=
AllowedOpenSSLVersions=1.1.1,1.0.2
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=60000
AuthenticationMethod=1
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=TLSv1.2,TLSv1.1,TLSv1
DataIntegrityLevel=1
DataIntegrityTypes=MD5,SHA1,SHA256,SHA384,SHA512
DefaultLongDataBufLen=1024
DescribeAtPrepare=0
EditionName=
EnableBulkLoad=0
EnableDescribeParam=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 175

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.

For more information, see [The Example Application](#) in *Progress DataDirect for ODBC Drivers Reference*.

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 Driver=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 57

[Connection Option Descriptions](#) on page 175

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, Linux, or macOS. See "Getting Started" 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
```

or, on macOS:

```
FILEDSN=/Library/ODBC/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

[Getting Started](#) on page 23

[DSN-less Connections](#) on page 63

[Configuring a Data Source in the System Information File](#) on page 58

UTF-16 Applications on UNIX and Linux

Because the DataDirect Driver Manager allows applications to use either UTF-8 or UTF-16 Unicode encoding, applications written in UTF-16 for Windows platforms can also be used on UNIX and Linux platforms.

The Driver Manager assumes a default of UTF-8 applications; therefore, two things must occur for it to determine that the application is UTF-16:

- The definition of SQLWCHAR in the ODBC header files must be switched from "char *" to "short *". To do this, the application uses #define SQLWCHARSHORT.
- The application must set the encoding for the environment or connection using one of the following attributes. If your application passes UTF-8 encoded strings to some connections and UTF-16 encoded strings to other connections in the same environment, encoding should be set for the connection only; otherwise, either method can be used.
 - To configure the encoding for the environment, set the ODBC environment attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16, for example:

```
rc = SQLSetEnvAttr(*henv,
SQL_ATTR_APP_UNICODE_TYPE, (SQLPOINTER)SQL_DD_CP_UTF16, SQL_IS_INTEGER);
```

- To configure the encoding for the connection only, set the ODBC connection attribute SQL_ATTR_APP_UNICODE_TYPE to a value of SQL_DD_CP_UTF16. For example:

```
rc = SQLSetConnectAttr(hdbc, SQL_ATTR_APP_UNICODE_TYPE, SQL_DD_CP_UTF16,
SQL_IS_INTEGER);
```

Configuring the Product on macOS

This chapter contains specific information about using your driver in the macOS environment.

Installing the Driver Manager for macOS

Before you can use the driver, you must install and setup the iODBC Driver Manager, version 3.52.7 or higher, on your machine. iODBC is an open-source interface that manages data sources and loads DataDirect drivers for macOS applications. It is the most commonly used Driver Manager for macOS platforms and is included with some versions of the operating systems. For more information, refer to <http://www.iodbc.org/>.

The Test Loading Tool

After installing the product, the next step in preparing to use a driver is to test load it. This can be accomplished using the command-line based tools, ivtestlib (32-bit) and ddtestlib (64-bit), described in this section or the iODBC Administrator. For more information on testing using the iODBC Administrator, see "Data Source Configuration through a GUI (macOS)."

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

For example, if the driver is installed in `~/Library/Progress/DataDirect/ODBC_80_64bit/lib`, the following command attempts to load the Condstart localhide driver, where `xx` represents the version number of the driver:

```
ddtestlib ~/Library/Progress/DataDirect/ODBC_80_64bit/lib/ddoraxx.dylib
```

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

See "Version String Information" for details about version strings.

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

See also

[Data Source Configuration through a GUI \(macOS\)](#) on page 70

[Version String Information](#) on page 43

Data Source Configuration on macOS

In the macOS environment, system information files, called `odbc.ini`, are used to store data source information. Two versions of the file, one for storing User DSN information and one for System DSN information, are placed in separate library directories on your machine (See "Configuration Through the System Information File" for locations of the `odbc.ini` files). The `odbc.ini` files are plain text files that contain data source definitions. Data source definitions can be created or modified by editing the `odbc.ini` files using a text editor or the graphical user interface (GUI) provided by the iODBC Administrator. For instructions on configuring data sources through the `odbc.ini` file, see "Configuration through the System Information (odbc.ini) File". See "Data Source Configuration through a GUI (macOS)" for details on using a GUI.

See also

[Configuration Through the System Information \(odbc.ini\) File](#) on page 66

[Data Source Configuration through a GUI \(macOS\)](#) on page 70

Configuration Through the System Information (odbc.ini) File

To configure a data source manually, edit the `odbc.ini` file that corresponds to the type of data source you want to use. On macOS platforms, the iODBC driver manager places two `odbc.ini` files on your machine: one for storing User DSN information and another for storing System DSN information. The location of this file determines the type of DSNs defined in the file. The following directories contain the `odbc.ini` file for their respective DSN type:

For User DSNs: `/Users/user_name/Library/ODBC/`

For System DSNs: `/Library/ODBC/`

By default, the installer program creates a default data source entry for the driver in the User DSN `odbc.ini` file. To create a System DSN, you will need to manually create a data source entry for the driver in the correlating `odbc.ini` file. See "File Data Sources" for information on creating a File DSN.

To edit the file, navigate to the appropriate directory and open the `odbc.ini` file using 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. 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=/Library/Progress/DataDirect/ODBC_80_64bit
```

The `IANAAppCodePage` keyword defines the default value that all UNIX/Linux drivers use if individual data sources have not specified a different value. See "IANAAppCodePage" and "Code Page Values" for details. The default value is 4.

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 `/usr/local`, then the following line is written to the [ODBC] section of the default `odbc.ini`:

```
InstallDir=/usr/local
```

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.

See also

[File Data Sources](#) on page 73
[Connection Option Descriptions](#) on page 175
[Sample `odbcinst.ini` File](#) on page 73
[IANAAppCodePage](#) on page 224
[Code Page Values](#) on page 267
[DSN-less Connections](#) on page 72
[ODBC Trace](#) on page 163

Sample Default odbc.ini File

The following is a sample `odbc.ini` file. 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.

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

[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ddora28.dylib
AccountingInfo=
Action=
AllowedOpenSSLVersions=1.1.1,1.0.2
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=60000
AuthenticationMethod=1
CachedCursorLimit=32
CachedDescLimit=0
CatalogIncludesSynonyms=1
CatalogOptions=0
ClientHostName=
ClientID=
ClientUser=
ConnectionRetryCount=0
ConnectionRetryDelay=3
CredentialsWalletEntry=
CredentialsWalletPassword=
CredentialsWalletPath=
CryptoLibName=
CryptoProtocolVersion=TLSv1.2,TLSv1.1,TLSv1
DataIntegrityLevel=1
DataIntegrityTypes=MD5,SHA1,SHA256,SHA384,SHA512
DefaultLongDataBufLen=1024
DescribeAtPrepare=0
EditionName=
EnableDescribeParam=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=
InitializationString=
KeepAlive=0
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalancing=0
LOBPrefetchSize=4000
LocalTimeZoneOffset=
LockTimeOut=-1
LoginTimeout=15
LogonID=
Module=0
Password=
PortNumber=<Oracle_server_port>
ProcedureRetResults=0
```

```

ProgramID=
PRNGSeedFile=/dev/random
PRNGSeedSource=0
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

[ODBC File DSN]
DefaultDSNDir=

```

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

- **To modify a data source:**

- Using a text editor, open the `odbc.ini` file.
- 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 "Connection Option Descriptions" for other specific attribute values.
- After making all modifications, save the `odbc.ini` file and close the text editor.

Important: The "Connection Option Description" 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:**

- Using a text editor, open the `odbc.ini` file.
- Copy an appropriate existing default data source definition and paste it to another location in the file.
- 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]`.
- 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 "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 "Connection Option Descriptions" 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.

Data Source Configuration through a GUI (macOS)

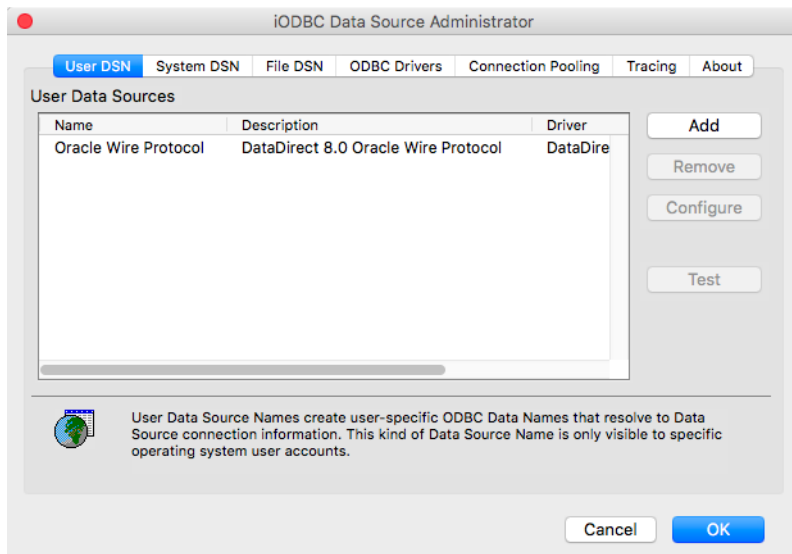
You must have the iODBC Administrator installed and set up on your machine. For information on downloading the iODBC Administrator, refer to <http://www.iodbc.org/>.

On macOS platforms, data sources are stored in the `odbc.ini` file. The iODBC Administrator allows you to create and modify these data sources using a graphical user interface, as described in this section.

To configure a data source:

1. Open `iODBC Administrator.app`.

The iODBC Data Source Administrator dialog box appears.



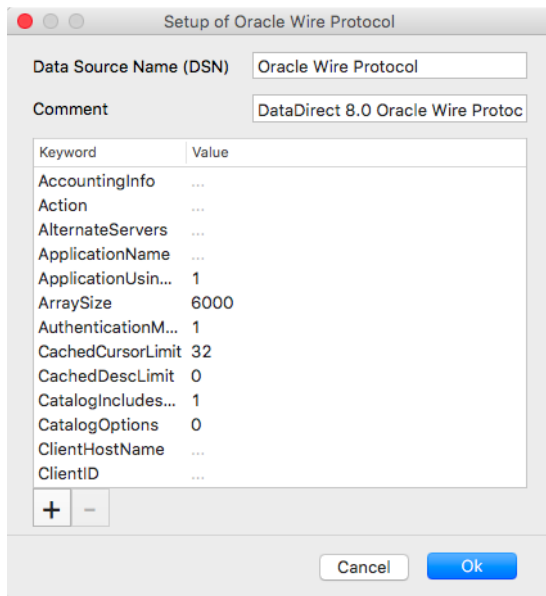
2. Click either the **User DSN**, **System DSN**, or **File DSN** tab to display a list of data sources.
 - **User DSN:** If you are configuring an existing user data source, select the appropriate data source name and click **Configure** to display the Setup dialog box.
If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the appropriate driver and click **Finish** to display the 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 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 Setup dialog box.
 - **File DSN:** If you are configuring an existing file data source, select the appropriate data source file and click **Configure** to display the Setup dialog box.

To configure a new file data source, click **Add** to display a list of installed drivers. Select the appropriate 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.

Note: If you want to set a default directory for File DSNs, select the directory from the Directories list; then, click **Set Dir**. The next time that you open the Administrator, it displays data source files from this directory.

See "Getting Started" for an explanation of different types of data sources.

3. The data source Setup dialog box appears. For existing data sources, the dialog is prepopulated with a list of the connection option attribute-value pairs currently stored in the `odbc.ini` file.



Configure the data source by adding and/or editing connection option attributes and values:

- To add a new connection option attribute, click the Add button . Then, in the new row, type the attribute name in the Keyword field and the desired valid value in the Value field.
- To configure an existing option, edit the value field that corresponds to the connection option attribute you want to modify.

For descriptions of connection option attributes and valid values, see "Connection Option Descriptions."

4. Click **OK** to save your changes and close the Setup dialog.
5. Optionally, test your connection. On the iODBC Data Source Administrator dialog, highlight your data source from the list; then, click **Test**.
6. A logon dialog box appears. Enter your username and password; then, click **Ok**. Note that the information you enter in the logon dialog box during a test connect is not saved.
7. Click **OK**. When you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can also override these defaults by connecting to the data source using a connection string with alternate values.

See also

[Getting Started](#) on page 23

[Connection Option Descriptions](#) on page 175

Tracing Using the iODBC Data Source Administrator

The Tracing tab allows you to trace calls to ODBC drivers and create a log of the traces for troubleshooting purposes.

To specify the path and name of the trace log file, type the path and name in the Trace File field or click **Browse** to select a log file. If no location is specified, the trace log resides in the working directory of the application you are using.

The iODBC Data Source Administrator ships with a trace library that is enabled by default. If you want to use a custom library instead, type the path and name of the library in the Custom trace library field or click **Browse** to select the library.

To enable tracing, indicate the frequency of tracing for the "When to trace" option on the Trace tab. If you select **All the time**, tracing continues until you disable it. Be sure to turn off tracing when you are finished reproducing the issue because tracing decreases the performance of your ODBC application.

After making changes on the Tracing tab, click **Apply** for them to take effect.

For a more complete discussion of tracing, see to "Diagnostic Tools."

When you are finished with the iODBC Administrator, click **OK** or **Cancel**. If you click **OK**, any changes you have made to the Trace tab are accepted and the Administrator closes.

See also

[Diagnostic Tools](#) 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. After you have configured your environment and data source, you can use the example application to test passing SQL statements.

Before you can begin testing, you will need to compile the application using the make files installed in the `/example` directory. You can then run the application by entering `example` from the Terminal and following 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.

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.

The product installer places a default version of this file in the `/Library/ODBC` 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 "Configuration Through the System Information (odbc.ini) File" for a description of the other keywords this section.

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

See also

[ODBCINST](#) on page 75

[Configuration Through the System Information \(odbc.ini\) File](#) on page 66

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.dylib
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help/OracleHelp
Setup=ODBCHOME/lib/ivora28.dylib
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
```

File Data Sources

The Driver Manager on macOS 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, regardless of platform. See "Getting Started" for a general description of ODBC data sources.

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
```

or, on macOS:

```
FILEDSN=/Library/ODBC/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 "Data Source Configuration on macOS" 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=/opt/odbc/filedsn/Oracleacct.dsn;UID=james;PWD=test01
```

See also

[Getting Started](#) on page 23

[DSN-less Connections](#) on page 72

[Configuration Through the System Information \(odbc.ini\) File](#) on page 66

Supported Character Encoding for macOS Applications

The iODBC Driver Manager allows macOS applications to use ASCII or UTF-32 character encoding. The Driver Manager automatically determines the encoding format used by the application based on whether the application calls ASCII or Unicode (wide or "W") functions.

Environment Variables

Most users do not need to configure their environment variables after installation. On macOS platforms, the location of system files, such as the `odbc.ini` and `odbcinst.ini` files, are standardized, making environment variable configuration unnecessary in most scenarios. Environment variables can be modified if necessary, but they must be set for each application that relies on the variable as well as the system.

The following procedure allows you to set the `ODBCINST` variable, if required by your application. For details on configuring additional environment variables, refer to the macOS documentation.

Note: To complete these procedures, you must 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 and affected directories.

ODBCINST

The installer program installs in the `/Library/ODBC` directory a default file, named `odbcinst.ini`, for use with DSN-less connections. However, if the user does not have access to the system library, the installer falls back to the `/Users/user_name/Library/ODBC` directory. See "DSN-less Connections" for an explanation of the `odbcinst.ini` file. The system administrator can choose to rename the file or move it to another location. In either scenario, 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 `/Users/john_smith/Library/myapp` in the Terminal, you would set this variable as follows:

```
setenv ODBCINST /Users/john_smith/Library/myapp/odbcinst.ini
```

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, for example `/Users/john_smith/`.

The Driver Manager searches for the location of the `odbcinst.ini` file as follows:

1. The Driver Manager checks the `ODBCINST` variable
2. The Driver Manager checks the `/Library/ODBC` and `/Users/user_name/Library/ODBC` directories for `.odbcinst.ini`

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

See also

[DSN-less Connections](#) on page 72

Data Source Configuration on Windows



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

The screenshot shows the 'ODBC Oracle Wire Protocol Driver Setup' dialog box with the 'General' tab selected. The 'Data Source Name' field contains 'Oracle Wire Protocol'. Below this are fields for 'Description', 'Standard Connection' (Host, Port Number, SID, Service Name, LDAP Distinguished Name), 'TNSNames Connection' (Server Name, TNSNames File), and 'Edition Name'. At the bottom are buttons for 'Test Connect', 'OK', 'Cancel', and 'Apply'.

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 207	Specifies the name of a data source in your Windows Registry or <code>odbc.ini</code> file. Default: None
Description on page 208	Specifies an optional long description of a data source. This description is not used as a runtime connection attribute, but does appear in the <code>ODBC.INI</code> section of the Registry and in the <code>odbc.ini</code> file. Default: None
Host on page 222	The name or the IP address of the server to which you want to connect. Default: None
Port Number on page 237	Specifies the port number of the server listener. Default: None
SID on page 252	The Oracle System Identifier that refers to the instance of Oracle running on the server. Default: None <hr/> Note: This option is mutually exclusive with the LDAP Distinguished Name, Service Name, Server Name, and TNSNames File options. <hr/> Note: If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the <code>ORCL</code> SID by default.
Service Name on page 251	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: <code>sales.us.acme.com</code> Default: None <hr/> Note: This option is mutually exclusive with the LDAP Distinguished Name, Service Name, Server Name, and TNSNames File options. <hr/> Note: If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the <code>ORCL</code> SID by default.

Connection Options: General	Description
<p>LDAP Distinguished Name on page 228</p>	<p>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 <code>orclNetDescString</code> attribute.</p> <hr/> <p>Note: This option is mutually exclusive with the Host, Port Number, SID, and Service Name options.</p> <hr/> <p>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.</p>
<p>Server Name on page 249</p>	<p>Specifies a net service name that exists in the <code>tnsnames.ora</code> file. The corresponding net service name entry in the <code>tnsnames.ora</code> file is used to obtain Host, Port Number, and Service Name or SID information.</p> <p>Default: None</p> <hr/> <p>Note: This option is mutually exclusive with the LDAP Distinguished Name, Host, Port Number, SID, and Service Name options.</p>
<p>TNSNames File on page 256</p>	<p>Specifies the name of the <code>tnsnames.ora</code> file.</p> <p>Default: None</p> <hr/> <p>Note: If no values are specified for the LDAP Distinguished Name, SID, Service Name, and TNSNames options, the driver attempts to connect to the <code>ORCL</code> SID by default.</p>
<p>Edition Name on page 209</p>	<p>The name of the Oracle edition the driver uses when establishing a connection. Oracle 11g 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 11g R2 and higher databases and tells the driver which edition of the schema objects to use.</p> <p>Default: None</p>

- 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 113

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 231	<p>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.</p> <p>Valid values are specified as offsets from GMT as follows: $(-)HH:MM$. For example, $-08:00$ equals GMT minus 8 hours.</p> <p>Default: None</p>

Connection Options: Advanced	Description
Default Buffer Size for Long/LOB Columns (in Kb) on page 207	<p>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.</p> <p>Default: 1024</p>
Application Using Threads on page 187	<p>Determines whether the driver works with applications using multiple ODBC threads.</p> <p>If enabled, the driver works with single-threaded and multi-threaded applications.</p> <p>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.</p> <p>Default: Enabled</p>
Describe at Prepare on page 208	<p>Determines whether the driver describes the SQL statement at prepare time.</p> <p>If enabled, the driver describes the SQL statement at prepare time.</p> <p>If disabled, the driver does not describe the SQL statement at prepare time.</p> <p>Default: Disabled</p>
Catalog Options on page 195	<p>Determines whether SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.</p> <p>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.</p> <p>If disabled, SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.</p> <p>Default: Disabled</p>
Support Binary XML on page 254	<p>Enables the driver to support XMLType with binary storage on servers running Oracle 12c and higher.</p> <p>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.</p> <p>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."</p> <p>Default: Disabled</p>

Connection Options: Advanced	Description
Enable SQLDescribeParam on page 213	<p>Determines whether the driver supports the SQLDescribeParam function, which allows an application to describe parameters in SQL statements and in stored procedure calls.</p> <p>If set to enabled, the driver supports SQLDescribeParam. If using Microsoft Remote Data Objects (RDO) to access data, you must use this value.</p> <p>If disabled, the driver does not support SQLDescribeParam and returns the error: <code>unimplemented function</code>.</p> <p>Default: Disabled</p>
Report Recycle Bin on page 248	<p>Determines whether support is provided for reporting objects that are in the Oracle Recycle Bin.</p> <p>If enabled, support is provided for reporting objects that are in the Oracle Recycle Bin.</p> <p>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\$.</p> <p>Default: Disabled</p>
Procedure Returns Results on page 244	<p>Determines whether the driver returns result sets from stored procedures/functions.</p> <p>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.</p> <p>If disabled, the driver does not return result sets from stored procedures.</p> <p>Default: Disabled</p>
Enable Server Result Cache on page 212	<p>Determines whether the driver sets the RESULT_CACHE_MODE session parameter to FORCE.</p> <p>If enabled, the driver sets the RESULT_CACHE_MODE session parameter to FORCE.</p> <p>If disabled, the driver does not sets the RESULT_CACHE_MODE session parameter.</p> <p>Default: Disabled</p>

Connection Options: Advanced	Description
Fetch TSWTZ as Timestamp on page 220	<p>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.</p> <p>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.</p> <p>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.</p> <p>Default: Disabled</p>
TCP Keep Alive on page 254	<p>Specifies whether the driver enables TCPKeepAlive. TCPKeepAlive maintains idle TCP connections by periodically passing packets between the client and server.</p> <p>If disabled, the driver does not enable TCPKeepAlive.</p> <p>If enabled, the driver enables TCPKeepAlive.</p> <p>Default: Disabled</p>
Timestamp Escape Mapping on page 255	<p>Determines how the driver maps Date, Time, and Timestamp literals.</p> <p>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.</p> <p>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.</p> <p>Default: 0 - Oracle Version Specific</p>

Connection Options: Advanced	Description
Report Codepage Conversion Errors on page 247	<p>Specifies how the driver handles code page conversion errors that occur when a character cannot be converted from one character set to another.</p> <p>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.</p> <p>If set to 1 - Return Error, the driver returns an error instead of substituting 0x1A for unconverted characters.</p> <p>If set to 2 - Return Warning, the driver substitutes 0x1A for each character that cannot be converted and returns a warning.</p> <p>Default: 0 - Ignore Errors</p>
Server Process Type on page 250	<p>Determines whether the connection is established using a shared or dedicated server process (dedicated thread on Windows).</p> <p>If set to 0 - Server Default, the driver uses the default server process set on the server.</p> <p>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.</p> <p>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.</p> <p>Default: 0 - Server Default</p>
Initialization String on page 226	<p>A SQL command that is issued immediately after connecting to the database to manage session settings.</p> <p>Default: None</p>
Login Timeout on page 233	<p>The number of seconds the driver waits for a connection to be established before returning control to the application and generating a timeout error.</p> <p>Default: 15</p>
Query Timeout on page 246	<p>The number of seconds for the default query timeout for all statements that are created by a connection.</p> <p>Default: 0 (the query does not time out.)</p>

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 79 in "Data Source Configuration through a GUI (Windows)." 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 85 allows you to specify security data source settings.
- [Performance Tab](#) on page 90 allows you to specify performance data source settings.
- [Failover Tab](#) on page 94 allows you to specify failover data source settings.
- [Pooling Tab](#) on page 97 allows you to specify connection pooling settings.
- [Bulk tab](#) on page 99 allows you to specify data source settings for DataDirect Bulk Load.
- [Client Monitoring Tab](#) on page 106 allows you to specify additional data source settings.
- [Advanced Security Tab](#) on page 108 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

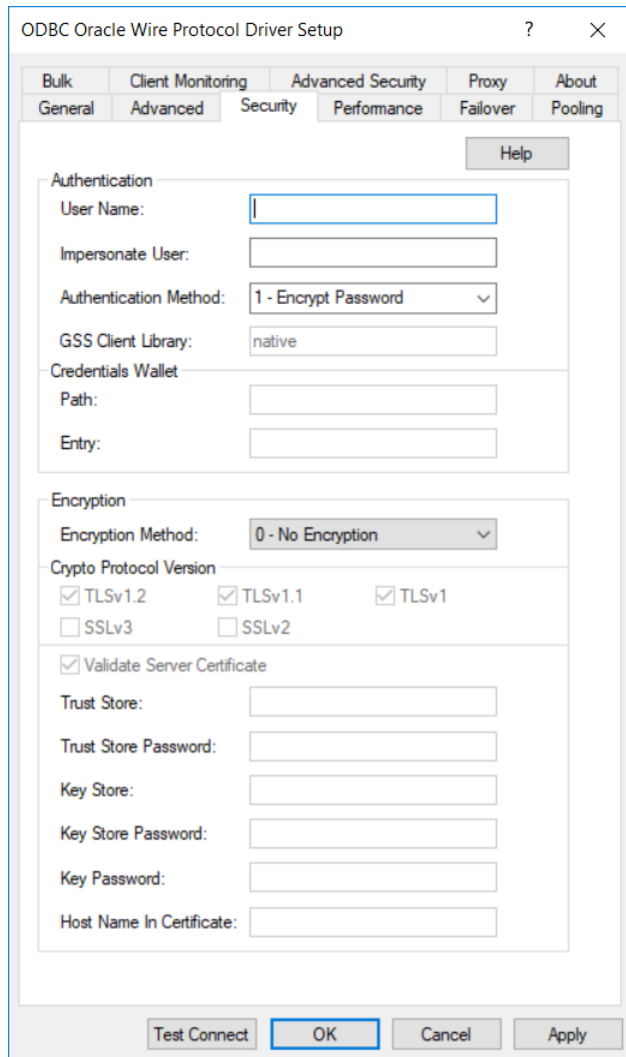
[Data Source Configuration on Windows](#) on page 75

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
<p>User Name on page 259</p>	<p>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.</p> <p>Default: None</p>

Connection Options: Security	Description
<p>Impersonate User on page 225</p>	<p>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.</p> <p>Default: None</p>
<p>Authentication Method on page 188</p>	<p>Specifies the method the driver uses to authenticate the user to the server when a connection is established.</p> <p>If set to 1 - Encrypt Password, the driver sends the user ID in clear text and an encrypted password to the server for authentication.</p> <p>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.</p> <p>If set to 4 - Kerberos Authentication, the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.</p> <p>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.</p> <p>If set to 6 - NTLM, the driver uses NTLMv1 authentication for Windows clients.</p> <p>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.</p> <p>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.</p> <p>If set to 14 - 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.</p> <p>Default: 1 - Encrypt Password</p>

Connection Options: Security	Description
Credentials Wallet Path on page 202	<p>Specifies the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. When Authentication Method is set to 14 - Wallet UID & PWD, the driver retrieves the database user name and password from this file.</p> <p>See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.</p>
Credentials Wallet Entry on page 201	<p>Specifies the string value used to identify database credential information stored in an Oracle Wallet. When Authentication Method is set to 14 - 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.</p> <p>See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.</p>
GSS Client Library on page 222	<p>The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).</p> <p>Default: <code>native</code></p>
Encryption Method on page 216	<p>The method the driver uses to encrypt data sent between the driver and the database server.</p> <p>If set to 0 - No Encryption, data is not encrypted.</p> <p>If set to 1 - SSL Auto, data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.</p> <p>Default: 0 - No Encryption</p>
Crypto Protocol Version on page 203	<p>A comma-separated list of the cryptographic protocols to use when 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.</p> <p>Default: <code>TLSv1.2, TLSv1.1, TLSv1</code></p>
Validate Server Certificate on page 259	<p>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.</p> <p>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.</p> <p>Default: Enabled</p>

Connection Options: Security	Description
Trust Store on page 257	The absolute path of the truststore file name to be used when SSL is enabled (<code>EncryptionMethod=1</code>) and server authentication is used. Default: None
Trust Store Password on page 258	Specifies the password that is used to access the truststore file when SSL is enabled (<code>EncryptionMethod=1</code>) and server authentication is used. Default: None
Key Store on page 227	The absolute path of the keystore file to be used when SSL is enabled (<code>EncryptionMethod=1</code>) and SSL client authentication is enabled on the database server. Default: None
Key Store Password on page 228	The password used to access the keystore file when SSL is enabled (<code>EncryptionMethod=1</code>) and SSL client authentication is enabled on the database server. Default: None
Key Password on page 226	The password used to access the individual keys in the keystore file when SSL is enabled (<code>Encryption Method=1</code>) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key. Default: None
Host Name In Certificate on page 223	A host name for certificate validation when SSL encryption is enabled (<code>EncryptionMethod=1</code>) and validation is enabled (<code>Validate Server Certificate=1</code>). Default: None

If you finished configuring your driver, proceed to Step 6 on page 79 in "Data Source Configuration through a GUI (Windows)." 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 133

[Oracle Wallet Password Store](#) on page 136

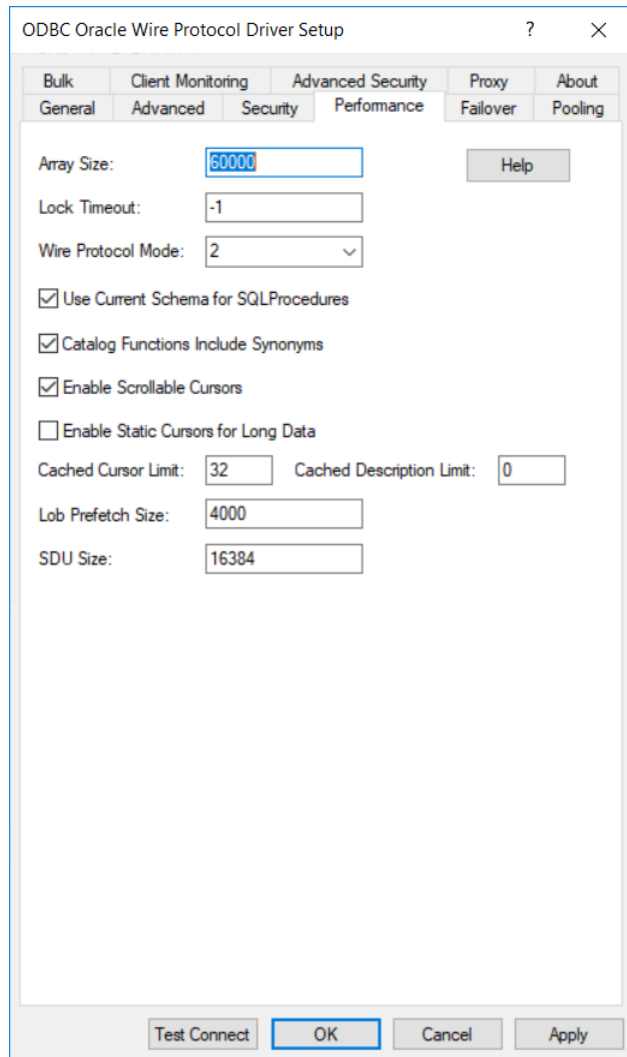
[Oracle Wallet SSL Authentication](#) on page 135

[Data Source Configuration on Windows](#) on page 75

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 188	<p>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.</p> <p>Default: 60000</p>
Lock Timeout on page 232	<p>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.</p> <p>If set to -1, the server waits indefinitely for the lock to be released.</p> <p>If set to 0, the server generates an error immediately and does not wait for the lock to time out.</p> <p>If set to x, the server waits for the specified number of seconds for the lock to be released.</p> <p>Default: -1</p>
Wire Protocol Mode on page 261	<p>Specifies whether the driver optimizes network traffic to the Oracle server.</p> <p>If set to 1, the driver operates in normal wire protocol mode without optimizing network traffic.</p> <p>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.</p> <p>Default: 2</p>
Use Current Schema for SQLProcedures on page 258	<p>When enabled, the call for SQLProcedures is optimized, but only procedures owned by the current user are returned.</p> <p>When disabled, the driver does not limit the procedures returned.</p> <p>Default: Enabled</p>
Catalog Functions Include Synonyms on page 194	<p>Determines whether synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.</p> <p>If enabled, synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.</p> <p>If disabled, synonyms are excluded (a non-standard behavior) and performance is thereby improved.</p> <p>Default: Enabled</p>

Connection Options: Performance	Description
Enable Scrollable Cursors on page 212	<p>Determines whether scrollable cursors, both Keyset and Static, are enabled for the data source.</p> <p>If set to enabled, scrollable cursors are enabled for the data source.</p> <p>If set to disabled, scrollable cursors are not enabled.</p> <p>Default: Enabled</p>
Enable Static Cursors for Long Data on page 214	<p>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.</p> <p>If enabled, the driver supports Long columns when using a static cursor.</p> <p>If disable, the driver does not support Long columns when using a static cursor.</p> <p>Default: Disabled</p>
Cached Cursor Limit on page 193	<p>Specifies the number of Oracle Cursor Identifiers that the driver stores in cache. A Cursor Identifier is needed for each concurrent open Select statement.</p> <p>Default: 32</p>
Cached Description Limit on page 193	<p>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.</p> <p>Default: 0</p>

Connection Options: Performance	Description
LOB Prefetch Size on page 231	<p>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.</p> <p>If set to -1, the property is disabled.</p> <p>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.</p> <p>If set to <i>x</i>, 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.</p> <p>Default: 4000</p>
SDU Size on page 248	<p>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.</p> <p>Default: 16384</p>

If you finished configuring your driver, proceed to [Step 6](#) on page 79 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.

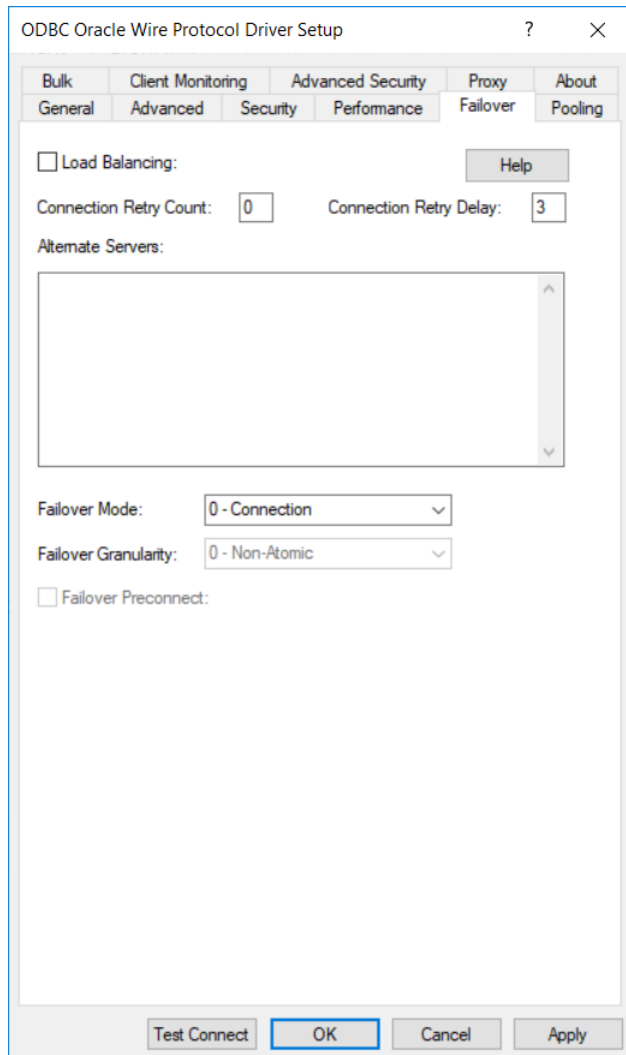
See also

[Data Source Configuration on Windows](#) on page 75

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 229	<p>Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate).</p> <p>If enabled, the driver uses client load balancing and attempts to connect to the database servers (primary and alternate servers) in random order.</p> <p>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).</p> <p>Default: Disabled</p>

Connection Options: Failover	Description
Connection Retry Count on page 199	<p>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.</p> <p>Default: 0</p>
Connection Retry Delay on page 200	<p>Specifies the number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer.</p> <p>If set to 0, there is no delay between retries.</p> <p>If set to <i>x</i>, the driver waits the specified number of seconds between connection retry attempts.</p> <p>Default: 3</p>
Alternate Servers on page 186	<p>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".</p> <p>Default: None</p>
Failover Mode on page 219	<p>Specifies the type of failover method the driver uses.</p> <p>If set to 0 - Connection, the driver provides failover protection for new connections only.</p> <p>If set to 1 - Extended Connection, the driver provides failover protection for new and lost connections, but not any work in progress.</p> <p>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.</p>

Connection Options: Failover	Description
Failover Granularity on page 218	<p>Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection.</p> <p>If set to 0 - Non-Atomic, the driver continues with the failover process and posts any errors on the statement on which they occur.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Default: 0 - Non-Atomic</p>
Failover Preconnect on page 220	<p>Specifies whether the driver tries to connect to the primary and an alternate server at the same time.</p> <p>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.</p> <p>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.</p> <p>Default: Disabled</p>

If you finished configuring your driver, proceed to Step 6 on page 79 in "Data Source Configuration through a GUI (Windows)." 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 123

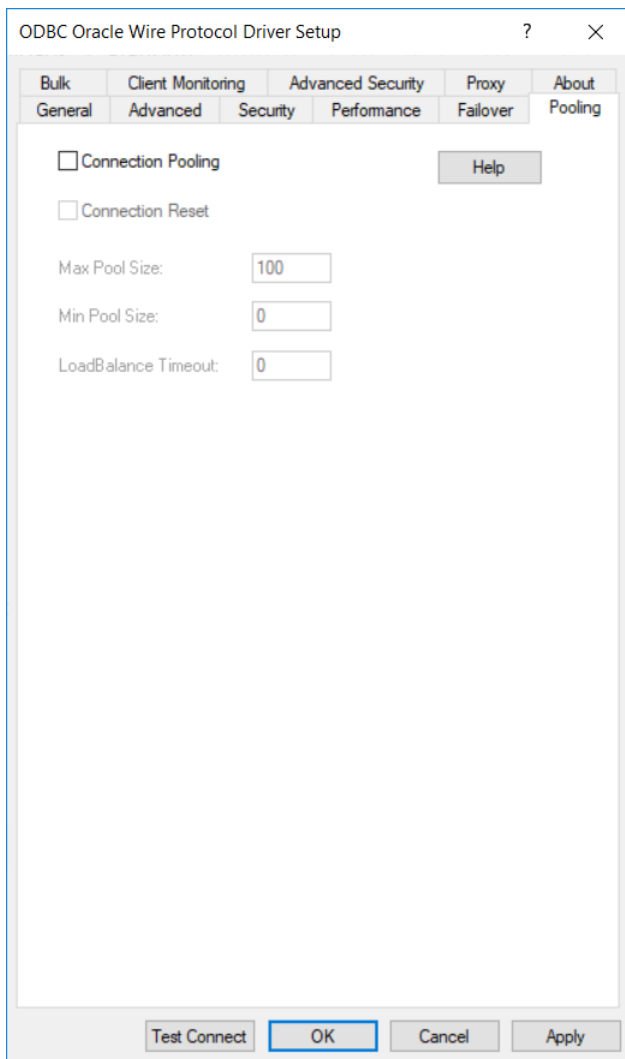
[Alternate Servers](#) on page 186

[Data Source Configuration on Windows](#) on page 75

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 198	<p>Specifies whether to use the driver's connection pooling.</p> <p>If enabled, the driver uses connection pooling.</p> <p>If disabled, the driver does not use connection pooling.</p> <p>Default: Disabled</p>
Connection Reset on page 198	<p>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.</p> <p>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.</p> <p>If disabled, the state of connections is not reset.</p> <p>Default: Disabled</p>
Max Pool Size on page 234	<p>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.</p> <p>Default: 100</p>
Min Pool Size on page 234	<p>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.</p> <p>Default: 0 (no connections are opened in addition to the current existing connection.)</p>
LoadBalance Timeout on page 230	<p>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.</p> <p>Default: 0 (inactive connections are kept open.)</p>

If you finished configuring your driver, proceed to Step 6 on page 79 in "Data Source Configuration through a GUI (Windows)." 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 145

[Data Source Configuration on Windows](#) on page 75

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

ODBC Oracle Wire Protocol Driver Setup

General Advanced Security Performance Failover Pooling
Bulk Client Monitoring Advanced Security Proxy About

Help

Enable Bulk Load

Bulk Options

No Index Errors

Field Delimiter: Record Delimiter:

Bulk Binary Threshold (KB): Batch Size:

Bulk Character Threshold (KB):

Load Table ...

Verify ...

Export Table ...

Test Connect OK Cancel Apply

Connection Options: Bulk	Description
Enable Bulk Load on page 210	<p>Specifies the bulk load method.</p> <p>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.</p> <p>If disabled, the driver uses standard parameter arrays.</p> <p>Default: Disabled</p>
No Index Errors	<p>Toggles options for the bulk load process.</p> <p>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.</p> <p>If disabled, the bulk load operation continues even if a value that would cause an index to be invalidated is loaded.</p> <p>Default: Disabled</p>
Field Delimiter on page 221	<p>Specifies the character that the driver will use to delimit the field entries in a bulk load data file.</p> <p>Default: None</p>
Record Delimiter on page 246	<p>Specifies the character that the driver will use to delimit the record entries in a bulk load data file.</p> <p>Default: None</p>
Bulk Binary Threshold on page 190	<p>The maximum size, in KB, of binary data that is exported to the bulk data file.</p> <p>If set to -1, all binary data, regardless of size, is written to the bulk data file, not to an external file.</p> <p>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.</p> <p>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.</p> <p>Default: None</p>

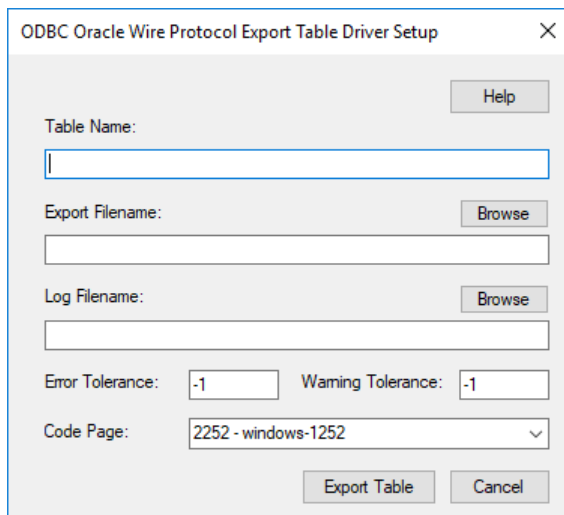
Connection Options: Bulk	Description
Batch Size on page 189	The number of rows that the driver sends to the database at a time during bulk operations. This value applies to all methods of bulk loading. Default: 1024
Bulk Character Threshold on page 191	The maximum size, in KB, of character data that 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/macOS, 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**.

- 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**.

- 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 79 in "Data Source Configuration through a GUI (Windows)." 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 149

[External Overflow Files](#) on page 156

[Character Set Conversions](#) on page 156

[Verification of the Bulk Load Configuration File](#) on page 154

[Data Source Configuration on Windows](#) on page 75

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 183	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 184	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 186	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 195	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 196	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 197	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 235	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 245	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 79 in "Data Source Configuration through a GUI (Windows)." 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

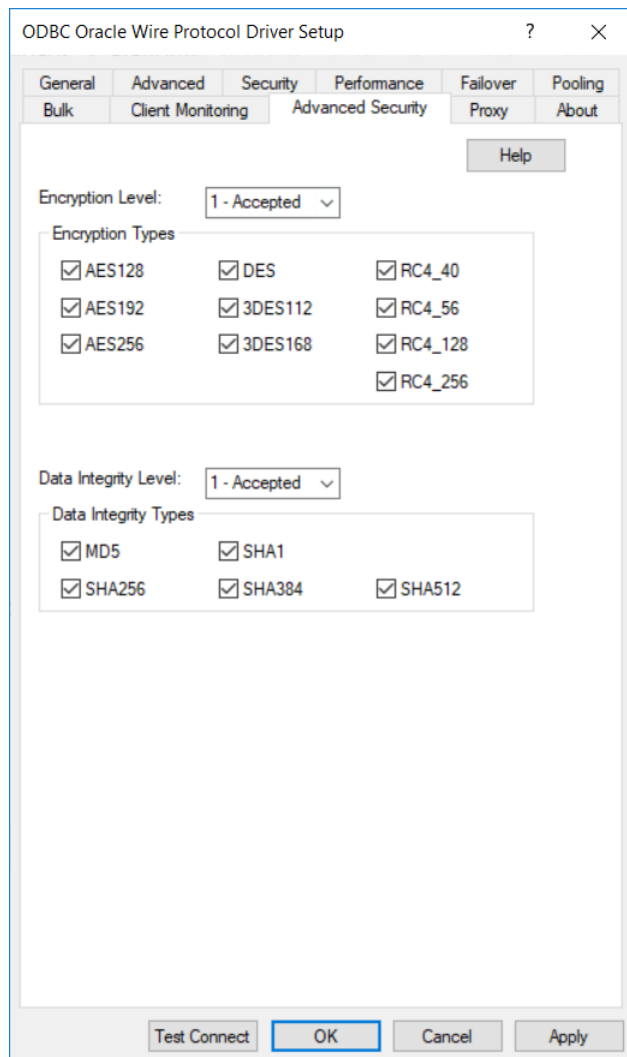
[Using Client Information](#) on page 132

[Data Source Configuration on Windows](#) on page 75

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 table; then, click **Apply**. The fields are optional unless otherwise noted.

Figure 12: Advanced Security tab



Connection Options: Advanced Security	Description
Encryption Level on page 215	<p>Specifies a preference on whether to use encryption on data being sent between the driver and the database server.</p> <p>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.</p> <p>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.</p> <p>If set to 2 - Requested, data sent between the driver and the database server is encrypted and decrypted if the database server permits it.</p> <p>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.</p> <p>Default: 1 - Accepted</p>
Encryption Types on page 217	<p>Specifies the encryption algorithms to use if Oracle Advanced Security encryption is enabled using the Encryption Level connection property.</p> <p>Default: All listed encryption algorithms are selected.</p>

Connection Options: Advanced Security	Description
Data Integrity Level on page 205	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>See "Encryption and Data Integrity" for more information.</p> <p>Default: 1 - Accepted</p>
Data Integrity Types on page 206	<p>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.</p> <p>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.</p> <p>Default: MD5, SHA1, SHA256, SHA384, SHA512</p>

If you finished configuring your driver, proceed to Step 6 on page 79 in "Data Source Configuration through a GUI (Windows)." 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 141

[Data Source Configuration on Windows](#) on page 75

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 13: Proxy tab

The screenshot shows the 'ODBC Oracle Wire Protocol Driver Setup' dialog box with the 'Proxy' tab selected. The dialog has a title bar with a question mark and a close button. Below the title bar are several tabs: 'General', 'Advanced', 'Security', 'Performance', 'Failover', 'Pooling', 'Bulk', 'Client Monitoring', 'Advanced Security', 'Proxy', and 'About'. The 'Proxy' tab is active, showing a 'Help' button and five input fields: 'Proxy Mode' (a dropdown menu set to '0 - NONE'), 'Proxy Host', 'Proxy Port', 'Proxy User', and 'Proxy Password'. At the bottom of the dialog are four buttons: 'Test Connect', 'OK', 'Cancel', and 'Apply'.

Connection Options: Advanced Security	Description
Proxy Mode on page 238	<p>Determines whether the driver connects to an endpoint through an HTTP proxy server.</p> <p>If set to 0 - NONE, the driver connects directly to the endpoint specified by the Host Name connection option.</p> <p>If set to 1 - HTTP, the driver connects to the endpoint through the HTTP proxy server specified by the ProxyHost connection option.</p> <p>Default: 0 - None</p>
Proxy Host on page 237	<p>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.</p> <p>Default: Empty string</p>
Proxy Port on page 240	<p>Specifies the port number where the Proxy Server is listening for HTTP requests.</p> <p>Default: 0</p>
Proxy User on page 241	<p>Specifies the user name needed to connect to the Proxy Server.</p> <p>Default: Empty string</p>
Proxy Password on page 239	<p>Specifies the password needed to connect to the Proxy Server.</p> <p>Default: Empty string</p>

If you finished configuring your driver, proceed to Step 6 on page 79 in "Data Source Configuration through a GUI (Windows)." 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 119

[Data Source Configuration on Windows](#) on page 75

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The logon dialog is not currently supported on macOS platforms. 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 175

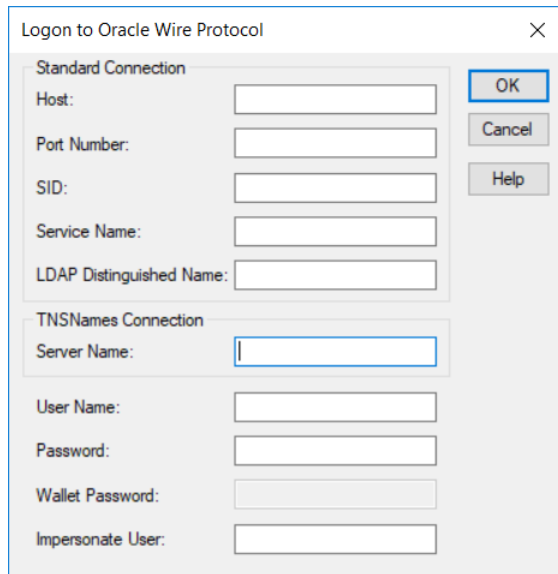
Using a Logon Dialog Box



Note: The logon dialog box is not currently supported on macOS platforms. When connecting on macOS platforms, all required information must be provided by the connection string or DSN.

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 14: 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 115.

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. See "Using IP Addresses" for details concerning these formats.
 - 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 115

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=14`), 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 8 on page 115
6. If required, type your Oracle user name.
7. If required, type your Oracle password.
8. 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.
9. 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 175

[Using IP Addresses](#) on page 52

[OS Authentication](#) on page 53

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. See "Designing ODBC Applications for Performance Optimization" for details.

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 11g 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.

See also

[Designing ODBC Applications for Performance Optimization](#) on page 295

Using LDAP



Supported on Windows, UNIX, and Linux only.

LDAP (Lightweight Directory Access Protocol) is a directory information service that allows you to centrally store information and share it across an IP network. In an LDAP service, information is stored in objects called entries, which can contain a variety of data—including connection information. LDAP entries are often used to store connection information because data storage is centralized, thereby simplifying maintenance when changes occur. The driver supports retrieving basic connection information from an LDAP entry, including:

- Oracle server name and portOracle System Identifier (SID) or Oracle service name
- Server process type (shared or dedicated)
- Failover instructions
- Client load balancing instructions
- SSL encryption instructions

To use the driver with LDAP, configure the following connection options:

- LDAP Distinguished Name (LDAPDistinguishedName): 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
```

- Host (HostName): Specify the name or IP address of the LDAP directory server.
- Port Number (PortNumber): Specify the port number listener of the LDAP directory server.

When attempting to connect, the driver retrieves connection information from the orclNetDescString attribute in the LDAP entry specified by the LDAP Distinguished Name option. The following is an example of an orclNetDescString value:

```
(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP) (HOST = host_name) (PORT =1521))) (CONNECT_DATA = (SID = ORCL)))
```

See also

[LDAP Distinguished Name](#) on page 228

[Host](#) on page 222

[Port Number](#) on page 237

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[Service Name](#) on page 251[SID](#) on page 252[Host](#) on page 222[Port Number](#) on page 237[Proxy Mode](#) on page 238[Proxy Host](#) on page 237[Proxy Port](#) on page 240[Proxy User](#) on page 241[Proxy Password](#) on page 239

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 `TNSNamesFile` (`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 `ServerName` (`ServerName`) option to specify the net service name for the Oracle Connection Manager 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 249

[TNSNames File](#) on page 256

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/macOS), 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 your 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.

See also

[Failover Mode](#) on page 219

[Connection Failover](#) on page 124

[Extended Connection Failover](#) on page 125

[Select Connection Failover](#) on page 126

[Alternate Servers](#) on page 186

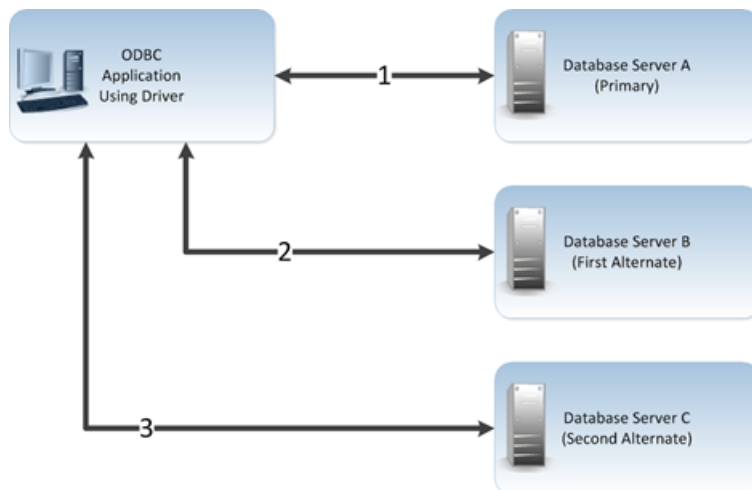
[Guidelines for Primary and Alternate Servers](#) on page 127

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 128

[Configuring Failover-Related Options](#) on page 128

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 124

[Failover Granularity](#) on page 218

[Failover Preconnect](#) on page 220

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 124

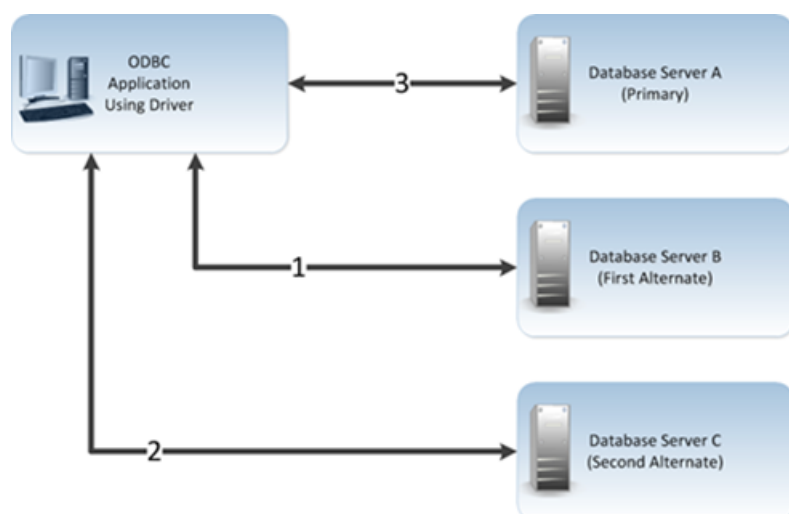
[Extended Connection Failover](#) on page 125

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 229

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 199

[Connection Retry Delay](#) on page 200

[Configuring Failover-Related Options](#) on page 128

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 3: Summary: Failover and Related Connection Options

Option	Characteristic
Alternate Servers (AlternateServers) (See step 1 on page 130)	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 131)	Number of times the driver retries the primary database server, and if specified, alternate servers until a successful connection is established. Default: 0

Option	Characteristic
<p>Connection Retry Delay (ConnectionRetryDelay)</p> <p>(See step 6 on page 131)</p>	<p>Wait interval, in seconds, between connection retry attempts when the Connection Retry Count option is set to a positive integer.</p> <p>Default: 3</p>
<p>Failover Granularity (FailoverGranularity)</p> <p>(See step 3 on page 130)</p>	<p>Determines whether the driver fails the entire failover process or continues with the process if errors occur while trying to reestablish a lost connection.</p> <p>If set to 0 (Non-Atomic), the driver continues with the failover process and posts any errors on the statement on which they occur.</p> <p>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.</p> <p>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.</p> <p>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).</p> <p>Default: 0 (Non-Atomic)</p>
<p>Failover Mode (FailoverMode)</p> <p>(See step 2 on page 130)</p>	<p>Specifies the type of failover method the driver uses.</p> <p>If set to 0 (Connection), the driver provides failover protection for new connections only.</p> <p>If set to 1 (Extended Connection), the driver provides failover protection for new and lost connections, but not any work in progress.</p> <p>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.</p> <p>Default: 0 (Connection)</p>

Option	Characteristic
<p>Failover Preconnect (FailoverPreconnect)</p> <p>(See step 4 on page 130)</p>	<p>Specifies whether the driver tries to connect to the primary and an alternate server at the same time.</p> <p>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.</p> <p>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.</p> <p>Default: 0 (Disabled)</p>
<p>Load Balancing (LoadBalancing)</p> <p>(See step 7 on page 131)</p>	<p>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.</p> <p>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.</p> <p>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).</p> <p>Default: 0 (Disabled)</p>

- 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).
- Choose a failover method by setting the Failover Mode connection option. The default method is Connection (`FailoverMode=0`).
- 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`).
- 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 175

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 driver
...
AlternateServers=(HostName=AccountingOracleServer: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.

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 11g 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 132

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 4: 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.

Option	Description	Location
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.
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 175

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.

For current information, refer to the security matrix on the Progress DataDirect Web site:

[Progress DataDirect Security Support Matrix](#)

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 5: Kerberos Authentication Requirements for the Oracle Wire Protocol Driver

Component	Requirements
Database server	<p>The database server must be administered by the same domain controller that administers the client and must be running one of the following databases:</p> <ul style="list-style-type: none"> • Oracle 12c (R1 and R2) • Oracle 11g (R1 and R2) • Oracle 10g (R1 and R2) • Oracle 9i (R2) <p>In addition, Oracle Advanced Security is required.</p>
Kerberos server	<p>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:</p> <ul style="list-style-type: none"> • 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	<p>The client must be administered by the same domain controller that administers the database server.</p>

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, Linux, macOS 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 188

[Encryption Method](#) on page 216

[Key Store](#) on page 227

[Key Store Password](#) on page 228

[Trust Store](#) on page 257

[Trust Store Password](#) on page 258

[User Name](#) on page 259

[Password](#) on page 236

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 Credentials Wallet Entry (`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 14 (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=14`), 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 not 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 188

[Credentials Wallet Path](#) on page 202

[Credentials Wallet Entry](#) on page 201

[Wallet Password](#) on page 260

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

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

Data Encryption and Integrity

The driver supports the following types of data encryption:

- SSL
- Oracle Advanced Security

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

SSL Encryption

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

- TLS v1.2, TLS v1.1, TLS v1.0
- SSL v3, SSL v2

The encryption cipher suite defines the type of encryption that is used for any data exchanged through an 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. See "SSL Encryption Cypher Suites" for a list of encryption cipher suites supported by the driver.

The handshake involves the following types of authentication:

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

See also

[SSL Encryption Cipher Suites](#) on page 309

Certificates

SSL requires the use of a digitally-signed document, an x.509 standard certificate, for authentication and the secure exchange of data. The purpose of this certificate is to tie the public key contained in the certificate securely to the person/company that holds the corresponding private key. 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

SSL Server Authentication

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

Most truststores are password-protected. The driver must be able to locate the truststore and unlock the truststore with the appropriate password. Two connection string attributes are available to the driver to provide this information: `TrustStore` and `TrustStorePassword`. The value of `TrustStore` is a pathname that specifies the location of the truststore file. The value of `TrustStorePassword` is the password required to access the contents of the truststore.

Alternatively, you can configure the driver to trust any certificate sent by the server, even if the issuer is not a trusted CA. Allowing a driver to trust any certificate sent from the server is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

`ValidateServerCertificate`, another connection string attribute, allows the driver to accept any certificate returned from the server regardless of whether the issuer of the certificate is a trusted CA.

Finally, the connection string attribute, `HostNameInCertificate`, allows an additional method of server verification. When a value is specified for `HostNameInCertificate`, it must match the host name of the server, which has been established by the SSL administrator. This prevents malicious intervention between the client and the server and ensures that the driver is connecting to the server that was requested.

SSL Client Authentication

If the server is configured for 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 string attributes are available to the driver to provide this information: `KeyStore` and `KeyStorePassword`. The value of `KeyStore` is a pathname that specifies the location of the keystore file. The value of `KeyStorePassword` is the password required to access the keystore.

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

Designating an OpenSSL Library

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

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

File replacement

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

Windows:

- Latest version: `xxtls28.dll`
- 1.0.2 and earlier versions: `xxssl28.dll`

UNIX/Linux:

- Latest version: `libxxtls28.so [.sl]`
- 1.0.2 and earlier versions: `libxxssl28.so [.sl]`

Designating a library in the default directory

If you are using the default directory structure for the product, you can use the `AllowedOpenSSLVersions` option to designate a library. To use the `AllowedOpenSSLVersions` option, specify the version number of the library you want to load. For example, `AllowedOpenSSLVersions=1.0.2` loads the 1.0.2 version of OpenSSL library using the following naming convention and format:

- Windows: `install_dir\drivers\xxssl28.so [.sl]`
- UNIX/Linux: `install_dir/lib/libxxtls28.so [.sl]`

Note that this method works only with OpenSSL library files that match Progress's naming convention and relative installation location.

If you are using the GUI, this option is not exposed on the setup dialog. Instead, use the Extended Options field on the Advanced tab to configure this option. For more information, see "AllowedOpenSSLVersions."

Designating the absolute path to a library

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

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

See also

[AllowedOpenSSLVersions](#) on page 184

[CryptoLibName](#) on page 204

[SSLibName](#) on page 253

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 227

[Key Store Password](#) on page 228

[Trust Store](#) on page 257

[Trust Store Password](#) on page 258

Oracle Advanced Security

To enable support for 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 6: Summary: Authentication Connection Options

Option	Description
Authentication Method (AuthenticationMethod)	<p>Specifies the method the driver uses to authenticate the user to the server when a connection is established.</p> <p>If set to 1 (Encrypt Password), the driver sends the user ID in clear text and an encrypted password to the server for authentication.</p> <p>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.</p> <p>If set to 4 (Kerberos Authentication), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.</p> <p>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.</p> <p>If set to 6 (NTLM), the driver uses NTLMv1 authentication for Windows clients.</p> <p>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.</p> <p>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.</p> <p>If set to 14 (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.</p> <p>Default: 1 (Encrypt Password)</p>
Credentials Wallet Entry (CredentialsWalletEntry)	<p>Specifies the string value used to identify database credential information stored in an Oracle Wallet. When Authentication Method is set to 14 (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.</p> <p>See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.</p>
Wallet Password (CredentialsWalletPassword)	<p>Specifies the password used to access the Oracle Wallet in which your database credential information is stored. When Authentication Method is set to 14 (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.</p> <p>See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.</p>

Option	Description
Credentials Wallet Path (CredentialsWalletPath)	<p>Specifies the fully-qualified path to the Oracle Wallet file in which your database credential information is stored. When Authentication Method is set to 14 (Wallet UID & PWD), the driver retrieves the database user name and password from this file.</p> <p>See "Oracle Wallet Password Store" for a complete list of options and settings required for the Oracle Wallet Password Store feature.</p>
GSS Client Library (GSSClient)	<p>The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).</p> <p>Default: <code>native</code> (The driver uses the GSS client shipped with the operating system.)</p>
ImpersonateUser (ImpersonateUser)	<p>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.</p> <p>Default: None</p>
User Name (LogonID)	<p>The default user ID that is used to connect to your database.</p> <p>Default: None</p>

Table 7: Summary: Data Encryption Connection Options

Option	Description
AllowedOpenSSLVersions (AllowedOpenSSLVersions)	<p>Determines which version of the OpenSSL library file the driver uses for data encryption.</p> <p>Default: <code>1.1.1, 1.0.2</code></p>
Crypto Protocol Version (CryptoProtocolVersion)	<p>Specifies the cryptographic protocols to use when SSL is enabled using the Encryption Method connection option (<code>EncryptionMethod=1</code>).</p> <p>Default: <code>TLSv1.2, TLSv1.1, TLSv1</code></p>
CryptoLibName (CryptoLibName)	<p>The absolute path for the OpenSSL library file containing the cryptographic library to be used by the data source or connection when SSL is enabled. The cryptographic library contains the implementations of cryptographic algorithms the driver uses for data encryption.</p> <p>Default: Empty string</p>

Option	Description
Encryption Method (EncryptionMethod)	<p>The method the driver uses to encrypt data sent between the driver and the database server.</p> <p>If set to 0 (No Encryption), data is not encrypted.</p> <p>If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.</p> <p>Default: 0 (No Encryption)</p>
Host Name In Certificate (HostNameInCertificate)	<p>A host name for certificate validation when SSL encryption is enabled (EncryptionMethod=1) and validation is enabled (Validate Server Certificate=1).</p> <p>Default: None</p>
Key Password (KeyPassword)	<p>Specifies the password used to access the individual keys in the keystore file when SSL is enabled (EncryptionMethod=1) and SSL client authentication is enabled on the database server.</p> <p>Default: None</p>
Key Store (Keystore)	<p>The absolute path of the keystore file to be used when SSL is enabled (EncryptionMethod=1) and SSL client authentication is enabled on the database server.</p> <p>Default: None</p>
Key Store Password (KeystorePassword)	<p>The password used to access the keystore file when SSL is enabled (EncryptionMethod=1) and SSL client authentication is enabled on the database server.</p> <p>Default: None</p>
SSLlibName (SSLlibName)	<p>The absolute path for the OpenSSL library file containing the SSL library to be used by the data source or connection when SSL is enabled. The SSL library contains the implementations of SSL protocols the driver uses for data encryption.</p> <p>Default: Empty string</p>
Trust Store (Truststore)	<p>The absolute path of the truststore file to be used when SSL is enabled (EncryptionMethod=1) and server authentication is used.</p> <p>Default: None</p>

Option	Description
Trust Store Password (TruststorePassword)	Specifies the password that is used to access the truststore file when SSL is enabled (<code>EncryptionMethod=1</code>) 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 175

[Oracle Wallet SSL Authentication](#) on page 135

[Oracle Wallet Password Store](#) on page 136

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 148

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 8: 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 124

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 9: Pool Statistics Connection Attribute

Connection Attribute	Description
SQL_ATTR_POOL_INFO Value: SQL_GET_POOL_INFO	<p>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 <code>qesqlext.h</code>. For example:</p> <pre>SQLGetConnectAttr(hdbc, SQL_ATTR_POOL_INFO, PoolInfoStruct *, SQL_LEN_BINARY_ATTR(PoolInfoStruct), &len);</pre> <p>This is a read-only connection attribute. The driver returns an error if SQLSetConnectAttr (SQL_ATTR_POOL_INFO) is called.</p>

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 10: Summary: Connection Pooling Connection Options

Option	Characteristic
Connection Pooling (Pooling)	<p>Specifies whether to use the driver's connection pooling.</p> <p>If set to 1 (Enabled), the driver uses connection pooling.</p> <p>If set to 0 (Disabled), the driver does not use connection pooling.</p> <p>Default: 0 (Disabled)</p>
Connection Reset (ConnectionReset)	<p>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.</p> <p>If 0 (Disabled), the state of connections is not reset.</p> <p>Default: 0 (Disabled)</p>
Load Balance Timeout (LoadBalanceTimeout)	<p>An integer value to specify the amount of time, in seconds, to keep connections open in a connection pool.</p> <p>Default: 0</p>

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 175

Using DataDirect Bulk Load



Supported on Windows, UNIX, and Linux only.

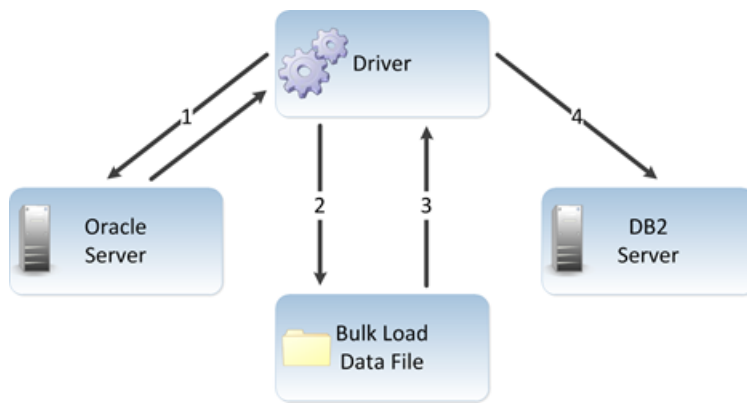
On Windows, UNIX, and Linux, the driver supports DataDirect Bulk Load when connected to Oracle databases version 9i R2 and higher. This feature allows your application to send large numbers of rows of data to a database. The driver sends the data to the database in a continuous stream instead of numerous 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 99

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 a `.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 153

[External Overflow Files](#) on page 156

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.

See also

[Bulk tab](#) on page 99

[DataDirect Bulk Load Functions](#) on page 317

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 151

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 99

Sample Applications

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

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

Character Set Conversions

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

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

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

[Code Page Values](#) on page 267

[External Overflow Files](#) on page 156

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 156

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)	<p>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.</p> <p>Default: 1024</p>
Bulk Binary Threshold (BulkBinaryThreshold)	<p>The maximum size, in KB, of binary data that is exported to the bulk data file.</p> <p>If set to -1, all binary data, regardless of size, is written to the bulk data file, not to an external file.</p> <p>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.</p> <p>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.</p> <p>Default: None</p>
Bulk Character Threshold (BulkCharacterThreshold)	<p>The maximum size, in KB, of character data that is exported to the bulk data file.</p> <p>If set to -1, all character data, regardless of size, is written to the bulk data file, not to an external file.</p> <p>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.</p> <p>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.</p> <p>Default: -1</p>

Connection Options: Bulk	Description
Bulk Load Options (BulkLoadOptions)	<p>Toggles options for the bulk load process.</p> <p>If set to <code>128</code> (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.</p> <p>If disabled, the bulk load operation continues even if a value that would cause an index to be invalidated is loaded.</p> <p>Default: Disabled</p>
Field Delimiter (BulkLoadFieldDelimiter)	<p>Specifies the character that the driver will use to delimit the field entries in a bulk load data file.</p> <p>Default: None</p>
Record Delimiter (BulkLoadRecordDelimiter)	<p>Specifies the character that the driver will use to delimit the record entries in a bulk load data file.</p> <p>Default: None</p>

See "Connection Option Descriptions" for details about configuring the options.

See also

[Connection Option Descriptions](#) on page 175

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)	<p>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.</p> <p>Default: 1024</p>
Enable Bulk Load (EnableBulkLoad)	<p>Specifies the bulk load method.</p> <p>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.</p> <p>If disabled, the driver uses standard parameter arrays.</p> <p>Default: Disabled</p>

See "Connection Option Descriptions" for details about configuring the options.

See also

[Connection Option Descriptions](#) on page 175

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
<i>StatementHandle</i>	The handle of the statement that contains the result set to persist as XML.
<i>Attribute</i>	SQL_PERSIST_AS_XML. This statement attribute can be found in the file qesqlxt.h, which is installed with the driver.
<i>ValuePtr</i>	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.
<i>StringLength</i>	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
```

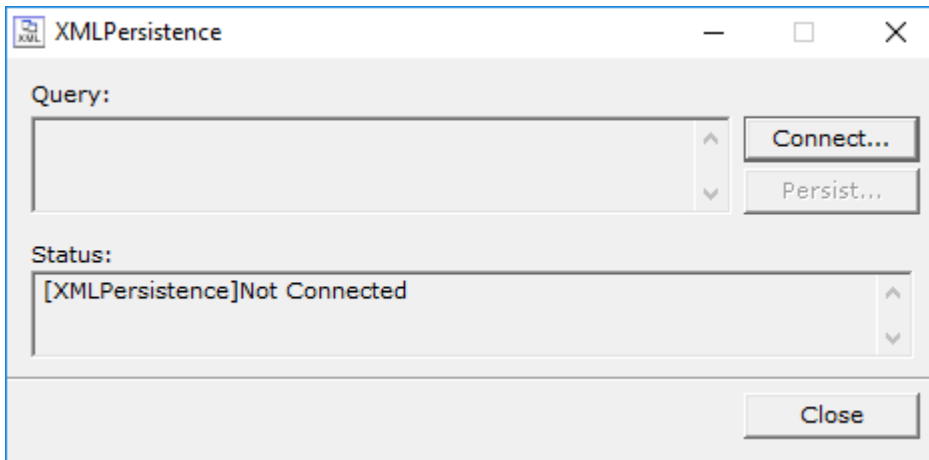
Using the Windows XML Persistence Demo Tool

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

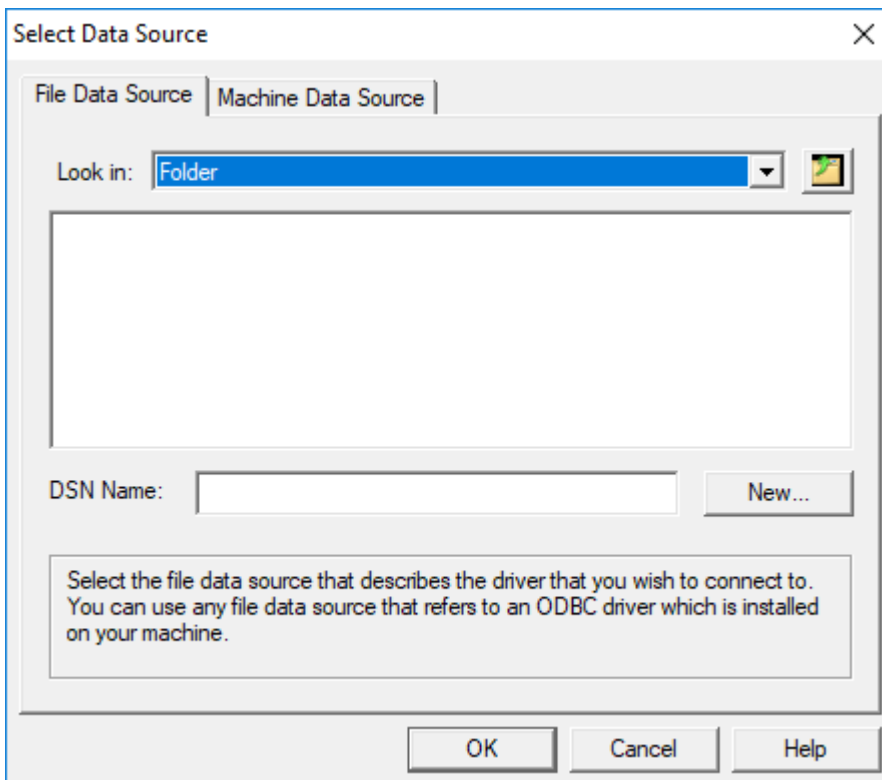
The tool has a graphical user interface and allows you to persist data as an XML data file.

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.

Troubleshooting

This part guides you through troubleshooting your Progress DataDirect *for* ODBC for Oracle Wire Protocol driver. It provides you with solutions to common problems and documents error messages that you may receive.

For details, see the following topics:

- [Diagnostic Tools](#)
- [Error Messages](#)
- [Troubleshooting](#)

Diagnostic Tools

This chapter discusses the diagnostic tools you use when configuring and troubleshooting your ODBC environment.

ODBC Trace

ODBC tracing allows you to trace calls to ODBC drivers and create a log of the traces.

Creating a Trace Log

Creating a trace log is particularly useful when you are troubleshooting an issue.

To create a trace log:

1. Enable tracing (see "Enabling Tracing" for more information).
2. Start the ODBC application and reproduce the issue.
3. Stop the application and turn off tracing.
4. Open the log file in a text editor and review the output to help you debug the problem.

For a complete explanation of tracing, refer to the following Progress DataDirect Knowledgebase document:

<http://knowledgebase.progress.com/articles/Article/3049>

See also

[Enabling Tracing](#) on page 164

Enabling Tracing

Progress DataDirect provides a tracing library that is enhanced to operate more efficiently, especially in production environments, where log files can rapidly grow in size. The DataDirect tracing library allows you to control the size and number of log files.

On Windows, you can enable tracing through the Tracing tab of the ODBC Data Source Administrator.

On UNIX and Linux, you can enable tracing by directly modifying the [ODBC] section in the system information (`odbc.ini`) file.

On macOS, you can also enable tracing through the Tracing tab of the iODBC Data Source Administrator.

Windows ODBC Administrator



On Windows, open the ODBC Data Source Administrator and select the Tracing tab. To specify the path and name of the trace log file, type the path and name in the Log File Path field or click **Browse** to select a log file. If no location is specified, the trace log resides in the working directory of the application you are using.

Click **Select DLL** in the Custom Trace DLL pane to select the DataDirect enhanced tracing library, `xxtrcy.dll`, where `xx` represents either `iv` (32-bit version) or `dd` (64-bit version), and `yy` represents the driver level number, for example, `ivtrc28.dll`. The library is installed in the `\Windows\System32` directory.

After making changes on the Tracing tab, click **Apply** for them to take effect.

Enable tracing by clicking **Start Tracing Now**. Tracing continues until you disable it by clicking **Stop Tracing Now**. Be sure to turn off tracing when you are finished reproducing the issue because tracing decreases the performance of your ODBC application.

When tracing is enabled, information is written to the following trace log files:

- Trace log file (`trace_filename.log`) in the specified directory.
- Trace information log file (`trace_filenameINFO.log`). This file is created in the same directory as the trace log file and logs the following SQLGetInfo information:
 - SQL_DBMS_NAME
 - SQL_DBMS_VER
 - SQL_DRIVER_NAME
 - SQL_DRIVER_VER
 - SQL_DEFAULT_TXN_ISOLATION

The DataDirect enhanced tracing library allows you to control the size and number of log files. The file size limit of the log file (in KB) is specified by the Windows Registry key ODBCTraceMaxFileSize. Once the size limit is reached, a new log file is created and logging continues in the new file until it reaches its file size limit, after which another log file is created, and so on.

The maximum number of files that can be created is specified by the Registry key ODBCTraceMaxNumFiles. Once the maximum number of log files is created, tracing 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. Subsequent files are named by appending sequential numbers, starting at 1 and incrementing by 1, to the end of the original file name, for example, `SQL1.LOG`, `SQL2.LOG`, and so on.

The default values of ODBCTraceMaxFileSize and ODBCTraceMaxNumFiles are 102400 KB and 10, respectively. To change these values, add or modify the keys in the following Windows Registry section:

```
[HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI\ODBC]
```

Warning: Do not edit the Registry unless you are an experienced user. Consult your system administrator if you have not edited the Registry before.

Edit each key using your values and close the Registry.

macOS iODBC Administrator



On macOS, you can enable tracing through the Tracing tab of the iODBC Data Source Administrator.

To specify the path and name of the trace log file, type the path and name in the Log file path field or click **Browse** to select a log file. If no location is specified, the trace log resides in the working directory of the application you are using.

The iODBC Data Source Administrator ships with a trace library that is enabled by default. If you want to use a custom library instead, type the path and name of the library in the Custom trace library field or click **Browse** to select the library.

To enable tracing, indicate the frequency of tracing for the "When to trace" option on the Trace tab. If you select **All the time**, tracing continues until you disable it. Be sure to turn off tracing when you are finished reproducing the issue because tracing decreases the performance of your ODBC application.

After making changes on the Tracing tab, click **Apply** for them to take effect.

The DataDirect enhanced tracing library gives you more control over tracing. See "System Information (odbc.ini) File" for a complete discussion of how to configure enhanced tracing.

See also

[System Information \(odbc.ini\) File](#) on page 166

System Information (odbc.ini) File



The [ODBC] section of the system information file includes several keywords that control tracing:

```
Trace=[0 | 1]
TraceFile=trace_filename
TraceDll=ODBCHOME/lib/xxtrcyy.zz
ODBCTraceMaxFileSize=file_size
ODBCTraceMaxNumFiles=file_number
TraceOptions=0
```

where:

```
Trace=[0 | 1]
```

Allows you to enable tracing by setting the value of Trace to 1. Disable tracing by setting the value to 0 (the default). Tracing continues until you disable it. Be sure to turn off tracing when you are finished reproducing the issue because tracing decreases the performance of your ODBC application.

```
TraceFile=trace_filename
```

Specifies the path and name of the trace log file. If no path is specified, the trace log resides in the working directory of the application you are using.

```
TraceDll=ODBCHOME/lib/xxtrcyy.zz
```

Specifies the library to use for tracing. The driver installation includes a DataDirect enhanced library to perform tracing, *xxtrcyy.zz*, where *xx* represents either *iv* (32-bit version) or *dd* (64-bit version), *yy* represents the driver level number, and *zz* represents either *so* or *sl*. For example, *ivtrc28.so* is the 32-bit version of the library. To use a custom shared library instead, enter the path and name of the library as the value for the TraceDll keyword.

The DataDirect enhanced tracing library allows you to control the size and number of log files with the ODBCTraceMaxFileSize and ODBCTraceMaxNumFiles keywords.

```
ODBCTraceMaxFileSize=file_size
```

The ODBCTraceMaxFileSize keyword 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 in the new file until it reaches the file size limit, after which another log file is created, and so on. The default is 102400.

```
ODBCTraceMaxNumFiles=file_number
```

The ODBCTraceMaxNumFiles keyword specifies the maximum number of log files that can be created. The default is 10. Once the maximum number of log files is created, tracing 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. Subsequent files are named by appending sequential numbers, starting at 1 and incrementing by 1, to the end of the original file name, for example, *odbctrace1.out*, *odbctrace2.out*, and so on.

```
TraceOptions=[0 | 1 | 2 | 3]
```

The ODBCTraceOptions keyword specifies whether to print the current timestamp, parent process ID, process ID, and thread ID for all ODBC functions to the output file. The default is 0.

- If set to 0, the driver uses standard ODBC tracing.
- If set to 1, the log file includes a timestamp on ENTRY and EXIT of each ODBC function.
- If set to 2, the log file prints a header on every line. By default, the header includes the parent process ID and process ID.
- If set to 3, both `TraceOptions=1` and `TraceOptions=2` are enabled. The header includes a timestamp as well as a parent process ID and process ID.

Example

In the following example of trace settings, tracing has been enabled, the name of the log file is `odbctrace.out`, the library for tracing is `ivtrc28.so`, the maximum size of the log file is 51200 KB, and the maximum number of log files is 8. Timestamp and other information is included in `odbctrace.out`.

```
Trace=1
TraceFile=ODBCHOME/lib/odbctrace.out
TraceDll=ODBCHOME/lib/ivtrc28.so
ODBCTraceMaxFileSize=51200
ODBCTraceMaxNumFiles=8
TraceOptions=3
```

The Test Loading Tool

Before using the test loading tool, be sure that your environment variables are set correctly. See "Environment Variable" for details about environment variables.

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

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

```
ivtestlib /opt/odbc/lib/ivoraxx.so
```

Note: On the HP-UX version, the full path to the driver must be specified for the tool. For other platforms, the full path is not required.

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

See "Version String Information" for details about version strings.

See also

[Environment Variables](#) on page 56

[Version String Information](#) on page 43

ODBC Test



On Windows, Microsoft® ships with its ODBC SDK an ODBC-enabled application, named ODBC Test, that you can use to test ODBC drivers and the ODBC Driver Manager. ODBC 3.52 includes both ANSI and Unicode-enabled versions of ODBC Test.

To use ODBC Test, you must understand the ODBC API, the C language, and SQL. For more information about ODBC Test, refer to the *Microsoft ODBC SDK Guide*.

iODBC Demo and iODBC Test



On macOS, the iODBC Driver Manager includes two sample applications, iODBC Demo and iODBC Test, that you can use to test ODBC drivers and the ODBC Driver Manager. iODBC Demo supports a graphical user interface to run tests, while iODBC Test employs a command-line interface. Both applications allow you to execute SQL statements against your environment, providing a quick means to test your connections, configurations, and setup. ANSI and Unicode-enabled versions of both applications are installed with the Driver Manager.

The Example Application

Progress DataDirect provides a simple C application, named example, that is useful for:

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

The example application is installed in the `/samples/example` subdirectory in the product installation directory. Refer to `example.txt` or `example64.txt` in the example directory for an explanation of how to build and use this application.

Other Tools

The Progress DataDirect Support Web site provides other diagnostic tools that you can download to assist you with troubleshooting. These tools are not shipped with the product. Refer to the Progress DataDirect Web page:

<https://www.progress.com/support/evaluation/download-resources/download-tools>

Progress DataDirect also provides a knowledgebase that is useful in troubleshooting problems. Refer to the Progress DataDirect Knowledgebase page:

<http://progresscustomersupport-survey.force.com/ConnectKB>

Error Messages

Error messages can be generated from:

- ODBC driver
- Database system
- ODBC driver manager

An error reported on an ODBC driver has the following format:

```
[vendor] [ODBC_component] message
```

where *ODBC_component* is the component in which the error occurred. For example, an error message from the Progress DataDirect for ODBC for Oracle Wire Protocol driver would look like this:

```
[DataDirect] [ODBC Oracle Wire Protocol Driver] Invalid precision specified.
```

If you receive this type of error, check the last ODBC call made by your application for possible problems or contact your ODBC application vendor.

An error that occurs in the data source includes the data store name, in the following format:

```
[vendor] [ODBC_component] [data_store] message
```

With this type of message, *ODBC_component* is the component that received the error specified by the data store. For example, you may receive the following message from an Oracle database:

```
[DataDirect] [ODBC Oracle Wire Protocol Driver] [Oracle] ORA-0919: specified length too long for CHAR column
```

This type of error is generated by the database system. Check your database system documentation for more information or consult your database administrator.

On UNIX and Linux the Driver Manager is provided by Progress DataDirect. For example, an error from the DataDirect Driver Manager might look like this:

```
[DataDirect][ODBC lib] String data code page conversion failed.
```

UNIX, Linux, and macOS error handling follows the X/Open XPG3 messaging catalog system. Localized error messages are stored in the subdirectory:

```
locale/localized_territory_directory/LC_MESSAGES
```

where *localized_territory_directory* depends on your language.

For instance, German localization files are stored in `locale/de/LC_MESSAGES`, where `de` is the locale for German.

If localized error messages are not available for your locale, then they will contain message numbers instead of text. For example:

```
[DataDirect] [ODBC 20101 driver] 30040
```



On Windows, the Microsoft Driver Manager is a DLL that establishes connections with drivers, submits requests to drivers, and returns results to applications. An error that occurs in the Driver Manager has the following format:

```
[vendor] [ODBC XXX] message
```

For example, an error from the Microsoft Driver Manager might look like this:

```
[Microsoft] [ODBC Driver Manager] Driver does not support this function
```

If you receive this type of error, consult the *Programmer's Reference* for the Microsoft ODBC Software Development Kit available from Microsoft.



On macOS, the iODBC Driver Manager establishes connections with drivers, submits requests to drivers, and returns results to applications. An error that occurs in the Driver Manager has the following format:

```
[vendor] [Driver Manager] message
```

For example, an error from the Microsoft Driver Manager might look like this:

```
[iODBC] [Driver Manager] Specified driver could not be loaded
```

If you receive this type of error, consult the iODBC documentation at <http://www.iodbc.org/>.

Troubleshooting

If you are having an issue while using your driver, first determine the type of issue that you are encountering:

- Setup/connection
- Performance
- Interoperability (ODBC application, ODBC driver, ODBC Driver Manager, or data source)
- Out-of-Memory

This chapter describes these three types of issues, provides some typical causes of the issues, lists some diagnostic tools that are useful to troubleshoot the issues, and, in some cases, explains possible actions you can take to resolve the issues.

Setup/Connection Issues

You are experiencing a setup/connection issue if you are encountering an error or hang while you are trying to make a database connection with the ODBC driver or are trying to configure the ODBC driver.

Some common errors that are returned by the ODBC driver if you are experiencing a setup/connection issue include:

- Specified driver could not be loaded.
- Data source name not found and no default driver specified.
- Cannot open shared library: libodbc.so.
- Unable to connect to destination.
- Invalid username/password; logon denied.

Troubleshooting the Issue

Some common reasons that setup/connection issues occur are:

- On Windows, UNIX, and Linux, the library path environment variable is not set correctly.

HP-UX ONLY:

- When setting the library path environment variable on HP-UX operating systems, specifying the parent directory is not required.
- You also must set the `LD_PRELOAD` environment variable to the fully qualified path of the `libjvm.so[sl]`.

The library path environment variable is:

32-bit Drivers

- `PATH` on Windows
- `LD_LIBRARY_PATH` on Solaris, Linux and HP-UX Itanium
- `SHLIB_PATH` on HP-UX PA_RISC
- `LIBPATH` on AIX

64-bit Drivers

- `PATH` on Windows
- `LD_LIBRARY_PATH` on Solaris, HP-UX Itanium, and Linux
- `LIBPATH` on AIX

- The database and/or listener are not started.
- The `ODBCINI` environment variable is not set correctly for the ODBC drivers on UNIX, Linux, or macOS.
- The ODBC driver's connection attributes are not set correctly in the system information file on UNIX, Linux, and macOS. See "Data Source Configuration on UNIX/Linux" or "Data Source Configuration for macOS" for more information. For example, the host name or port number are not correctly configured. See "Connection Option Descriptions" for a list of connection string attributes that are required for each driver to connect properly to the underlying database.

See "The Test Loading Tool" for information about a helpful diagnostic tool.

See also

[Data Source Configuration on UNIX/Linux](#) on page 58

[Data Source Configuration on macOS](#) on page 66

[Connection Option Descriptions](#) on page 175

[Configuring the Product on macOS](#) on page 65

[The Test Loading Tool](#) on page 167

Interoperability Issues

Interoperability issues can occur with a working ODBC application in any of the following ODBC components: ODBC application, ODBC driver, ODBC Driver Manager, and/or data source. See "What Is ODBC?" for more information about ODBC components.

For example, any of the following problems may occur because of an interoperability issue:

- SQL statements may fail to execute.
- Data may be returned/updated/deleted/inserted incorrectly.
- A hang or core dump may occur.

See also

[What Is ODBC?](#) on page 31

Troubleshooting the Issue

Isolate the component in which the issue is occurring. Is it an ODBC application, an ODBC driver, an ODBC Driver Manager, or a data source issue?

To troubleshoot the issue:

1. Test to see if your ODBC application is the source of the problem. To do this, replace your working ODBC application with a more simple application. If you can reproduce the issue, you know your ODBC application is **not** the cause.

UNIX[®]

On UNIX and Linux, you can use the example application that is shipped with your driver. See "The example Application" for details.



On Windows, you can use ODBC Test, which is part of the Microsoft ODBC SDK, or the example application that is shipped with your driver. See "ODBC Test" and "The example Application" for details.



Mac OS

On macOS, you can use iODBC Demo or iODBC Test, which are installed with the iODBC Administrator, or the example application that is shipped with your driver. See "iODBC Demo and iODBC Test" and "The example Application" for details.

2. Test to see if the data source is the source of the problem. To do this, use the native database tools that are provided by your database vendor.
3. If neither the ODBC application nor the data source is the source of your problem, troubleshoot the ODBC driver and the ODBC Driver Manager.

In this case, we recommend that you create an ODBC trace log to provide to Technical Support. See "ODBC Trace" for details.

See also

[ODBC Test](#) on page 168

[The Example Application](#) on page 168

[iODBC Demo and iODBC Test](#) on page 168

[ODBC Trace](#) on page 163

Performance Issues

Developing performance-oriented ODBC applications is not an easy task. You must be willing to change your application and test it to see if your changes helped performance. Microsoft's *ODBC Programmer's Reference* does not provide information about system performance. In addition, ODBC drivers and the ODBC Driver Manager do not return warnings when applications run inefficiently.

Some general guidelines for developing performance-oriented ODBC applications include:

- Use catalog functions appropriately.
- Retrieve only required data.
- Select functions that optimize performance.
- Manage connections and updates.

See "Designing ODBC Applications for Performance Optimization" for complete information.

See also

[Designing ODBC Applications for Performance Optimization](#) on page 295

Connection Option Descriptions

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

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

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

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

Table 11: Oracle Wire Protocol Attribute Names

Attribute (Short Name)	Default
AccountingInfo (AI)	None
Action (ACT)	None
AlternateServers (ASRV)	None
AllowedOpenSSLVersions (AOV)	latest
ApplicationName (AN)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	60000

Attribute (Short Name)	Default
AuthenticationMethod (AM)	1 (Encrypt Password)
BulkBinaryThreshold (BBT) (Windows, UNIX, and Linux only)	32
BulkCharacterThreshold (BCT) (Windows, UNIX, and Linux only)	-1
BulkLoadBatchSize (BLBS) (Windows, UNIX, and Linux only)	1024
BulkLoadFieldDelimiter (BLFD) (Windows, UNIX, and Linux only)	None
BulkLoadOptions (BLO) (Windows, UNIX, and Linux only)	0
BulkLoadRecordDelimiter (BLRD) (Windows, UNIX, and Linux only)	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) (Windows, UNIX, and Linux only)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CredentialsWalletEntry (CWE)	None
CredentialsWalletPassword (CWPWD)	None
CredentialsWalletPath (CWPATH)	None
CryptoProtocolVersion (CPV)	TLSv1.2, TLSv1.1, TLSv1
CryptoLibName (CLN)	Empty string
DataIntegrityLevel (DIL)	1 (Accepted)
DataIntegrityTypes (DIT)	MD5, SHA1, SHA256, SHA384, SHA512
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024

Attribute (Short Name)	Default
DescribeAtPrepare (DAP)	0 (Disabled)
Description (n/a)	None
EditionName (EN)	None
EnableBulkLoad (EBL) (Windows, UNIX, and Linux only)	0 (Disabled)
EnableDescribeParam (EDP)	0 (Disabled)
EnableNcharSupport (ENS)	None
Note: EnableNcharSupport has been deprecated.	
EnableScrollableCursors (ESC)	1 (Enabled)
EnableServerResultCache (ESRC)	0 (Disabled)
EnableStaticCursorsForLongData (ESCLD)	0 (Disabled)
EnableTimestampwithTimezone (ETWT)	None
Note: EnableTimestampwithTimezone has been deprecated.	
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.
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
GSSClient (GSSC)	native
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) (UNIX, Linux, macOS only)	4 (ISO 8559-1 Latin-1)

Attribute (Short Name)	Default
ImpersonateUser (IU)	None
InitializationString (IS)	None
KeepAlive (KA)	0 (Disabled)
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LDAPDistinguishedName (LDN) (Windows, UNIX, and Linux only)	None
LoadBalanceTimeout (LBT) (Windows, UNIX, and Linux only)	0
LoadBalancing (LB)	0 (Disabled)
LOBPrefetchSize (LPS)	4000
LocalTimezoneOffset (LTZO)	" " (Empty String)
LockTimeout (LTO)	-1
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS) (Windows, UNIX, and Linux only)	100
MinPoolSize (MNPS) (Windows, UNIX, and Linux only)	0
Module (MOD)	None
Password (PWD)	None
Pooling (POOL) (Windows, UNIX, and Linux only)	0 (Disabled)
PortNumber (PORT)	None
PRNGSeedFile (PSF) (UNIX, Linux, macOS only)	/dev/random
PRNGSeedSource (PSS) (UNIX, Linux, macOS only)	0 (File)
ProcedureRetResults (PRR)	0 (Disabled)
ProgramID (PID)	None
ProxyHost (PXHN) (Windows, UNIX, and Linux only)	Empty string
ProxyMode (PXM) (Windows, UNIX, and Linux only)	0 (NONE)

Attribute (Short Name)	Default
ProxyPassword (PXPW) (Windows, UNIX, and Linux only)	Empty string
ProxyPort (PXPT) (Windows, UNIX, and Linux only)	0
ProxyUser (PXU) (Windows, UNIX, and Linux only)	Empty string
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
ReportRecycleBin (RRB)	0 (Disabled)
SDUSize (SDU)	16384
ServerName (SRVR)	None
ServerType (ST)	0 (Server 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)	0 (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)

Attribute (Short Name)	Default
ValidateServerCertificate (VSC)	1 (Enabled)
WireProtocolMode (WPM)	2

For details, see the following topics:

- [Accounting Info](#)
- [Action](#)
- [AllowedOpenSSLVersions](#)
- [Alternate Servers](#)
- [Application Name](#)
- [Application Using Threads](#)
- [Array Size](#)
- [Authentication Method](#)
- [Batch Size](#)
- [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 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
 - 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 Distinguished Name
 - Load Balancing

- [LoadBalance Timeout](#)
- [LOB Prefetch Size](#)
- [Local Timezone Offset](#)
- [Lock Timeout](#)
- [Login Timeout](#)
- [Max Pool Size](#)
- [Min Pool Size](#)
- [Module](#)
- [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 115

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 10g 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 115

AllowedOpenSSLVersions

Attribute

AllowedOpenSSLVersions (AOV)

Purpose

Important: Version 1.0.2 of the OpenSSL library has reached the end of its product life cycle and is no longer receiving security updates. Best security practices dictate that you use the latest version of the library.

Determines which version of the OpenSSL library file the driver uses for data encryption. Although the latest version of the OpenSSL library is the most secure, some characteristics of the library can cause connections to certain databases to fail. This option allows you to continue using older versions of the OpenSSL library while you transition your environment to support the latest version.

Valid Values

`latest | openssl_version_number[,openssl_version_number]...`

where:

openssl_version_number

is the version number for the OpenSSL library file to be loaded by the driver, for example, 1.0.2. When more than one version is specified, the driver will first attempt to load the first version listed. If the driver is unable to locate and load this file, it will attempt to load the next version in the value. The driver currently supports versions 1.1.1 and 1.0.2. Refer to the installed readme for latest supported versions.

Behavior

If set to `latest`, the driver loads the latest installed version of the OpenSSL library file provided by Progress.

If set to `openssl_version_number`, the driver loads the specified version of the OpenSSL library file. This value is used to specify a version other than the latest.

Notes

- This option is ignored if OpenSSL library files are specified using the `CryptoLibName` and `SSLLibName` options.
- This option works only with OpenSSL library files provided by Progress and user supplied OpenSSL library files that match Progress's naming convention and installation location.
- This option works only for installations using the default directory structure.
- Consult your database administrator concerning the security settings of your server.

Default

`1.1.1,1.0.2`

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:

```
AllowedOpenSSLVersions=1.0.2
```

See also

- [Advanced Tab](#) on page 80

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 115

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 115

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

An integer from 1 to 4,294,967,296 (4 GB)

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 115

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 | 14

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 14 (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.

Notes

- When AuthenticationMethod is set to 14 (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 259
- [Password](#) on page 236
- [Oracle Wallet SSL Authentication](#) on page 135

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](#)

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 115

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 115

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 115

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 115

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 115

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 10g 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 115

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 115

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 115

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 115

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=14` (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=14` (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 188
- [Wallet Password](#) on page 260
- [Oracle Wallet Password Store](#) on page 136

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=14` (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

- 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 188
- [Wallet Password](#) on page 260
- [Oracle Wallet Password Store](#) on page 136

Crypto Protocol Version

Attribute

CryptoProtocolVersion (CPV)

Purpose

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

Valid Values

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

where:

```
cryptographic_protocol
```

is one of the following cryptographic protocols:

```
TLSv1.2 | TLSv1.1 | TLSv1 | SSLv3 | SSLv2
```

Caution: Good security practices recommend using TLSv1 or higher, due to known vulnerabilities in the SSLv2 and SSLv3 protocols.

Example

If your security environment is configured to use TLSv1.2 and TLSv1.1, specify the following values:

```
CryptoProtocolVersion=TLSv1.2, TLSv1.1
```

Notes

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

Default

```
TLSv1.2, TLSv1.1, TLSv1
```

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 SSL is enabled. The cryptographic 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_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

Notes

- The OpenSSL library files provided by Progress combine the cryptographic and 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.

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_80\drivers\OpenSSL\1.0.2d\ddssl28.dll;
```

See also

- [SSLibName](#) on page 253
- [Advanced Tab](#) on page 80

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 141
- [Performance Considerations](#) on page 115

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 205
- [Oracle Advanced Security](#) on page 141
- [Performance Considerations](#) on page 115

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 1024×2048 , or 2097152 (2 MB).

Notes

- This connection option can affect performance.

Default

1024

GUI Tab

[Advanced tab](#)

See also

[Performance Considerations](#) on page 115

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 115

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 11g 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 11g 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 115

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 9i and higher.

Default

None

See also

- [Advanced Tab](#) on page 80

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 115

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 11g 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 115

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 115

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 80

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 115

Encryption Method

Attribute

EncryptionMethod (EM)

Purpose

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

Valid Values

0 | 1

Behavior

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

If set to 1 (SSL), data is encrypted using the SSL protocols specified in the Crypto Protocol Version connection option.

Notes

- Consult your database administrator concerning the SSL settings of your Oracle server.
- This connection option can affect performance.

Default

0 (No Encryption)

GUI Tab[Security tab](#)**See also**[Performance Considerations](#) on page 115

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 115

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 10g 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

None

GUI Tab

[Bulk tab](#)

GSS Client Library

Attribute

GSSClient (GSSC)

Purpose

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

The driver uses the path defined by the PATH environment variable for loading the specified client library.

Valid Values

native | *client_library*

where:

client_library is a GSS client library installed on the client.

Behavior

If set to *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.

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. See "Using IP Addresses" for details about these formats.

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](#)

See Also

- [Using IP Addresses](#) on page 52

Host Name In Certificate

Attribute

HostNameInCertificate (HNIC)

Purpose

A host name for certificate validation when 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 SSL administrator for the correct value.

Behavior

If set to a *host_name*, the driver examines the subjectAltName values included in the certificate. If a dnsName value is present in the subjectAltName values, then the driver compares the value specified for Host Name In Certificate with the dnsName value. The connection succeeds if the values match. The connection fails if the Host Name In Certificate value does not match the dnsName value.

If no subjectAltName values exist or a dnsName value is not in the list of subjectAltName values, then the driver compares the value specified for Host Name In Certificate with the commonName part of the Subject name in the certificate. The commonName typically contains the host name of the machine for which the certificate was created. The connection succeeds if the values match. The connection fails if the Host Name In Certificate value does not match the commonName. If multiple commonName parts exist in the Subject name of the certificate, the connection succeeds if the Host Name In Certificate value matches any of the commonName parts.

If set to #*SERVERNAME*#, the driver compares the host server name specified as part of a data source or connection string to the dnsName or the commonName value.

Default

None

GUI Tab

[Security tab](#)

IANAAppCodePage



Supported on UNIX, Linux, and macOS 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 `qesql_ext.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 "Code Page Values." The value must match the database character encoding and the system locale.

Default

4 (ISO 8559-1 Latin-1)

GUI Tab

[Advanced tab](#)

See also

[Code Page Values](#) on page 267

[Internationalization, Localization, and Unicode](#) on page 283

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 SSL is enabled (`Encryption Method=0 | 1 | 3 | 4 | 5`) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values

key_password

where:

key_password

is the password of a key in the keystore.

Default

None

GUI Tab

[Security tab](#)

Key Store

Attribute

Keystore (KS)

Purpose

The absolute path of the keystore file to be used when SSL is enabled (`EncryptionMethod=1`) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request. If you do not specify a directory, the current directory is used.

Valid Values

keystore_directory

where:

keystore_directory

is the location of the keystore file.

Notes

- The keystore and truststore files can be the same file.

Default

None

GUI Tab

[Security tab](#)

Key Store Password

Attribute

KeystorePassword (KSP)

Purpose

The password used to access the keystore file when SSL is enabled (`EncryptionMethod=1`) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

Valid Values

keystore_password

where:

keystore_password

is the password of the keystore file.

Notes

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

Default

None

GUI Tab

[Security tab](#)

LDAP Distinguished Name



Supported on Windows, UNIX, and Linux only.

Attribute

LDAPDistinguishedName (LDN)

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 222
- [Port Number](#) on page 237
- [Using LDAP](#) on page 119

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 115

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 9i 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 115

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 115

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 115

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 10g 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 115

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

None

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. See "Using IP Addresses" for details about these formats.

Default

Empty string

Notes

- When proxy mode is disabled (`ProxyMode=0`), the Proxy Host option is ignored.

GUI Tab

[General tab](#)

See Also

- [Using IP Addresses](#) on page 52
- [Proxy Mode](#) on page 238
- [Proxy Password](#) on page 239
- [Proxy Port](#) on page 240
- [Proxy User](#) on page 241

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 237
- [Host](#) on page 222
- [Using IP Addresses](#) on page 52
- [Proxy Password](#) on page 239
- [Proxy Port](#) on page 240
- [Proxy User](#) on page 241

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

- [Using IP Addresses](#) on page 52
- [Proxy Host](#) on page 237
- [Proxy Mode](#) on page 238
- [Proxy Port](#) on page 240
- [Proxy User](#) on page 241

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

- [Using IP Addresses](#) on page 52
- [Proxy Host](#) on page 237
- [Proxy Mode](#) on page 238
- [Proxy Password](#) on page 239
- [Proxy User](#) on page 241

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

- [Using IP Addresses](#) on page 52
- [Proxy Host](#) on page 237
- [Proxy Mode](#) on page 238
- [Proxy Password](#) on page 239
- [Proxy Port](#) on page 240

PRNGSeedFile



Supported on UNIX, Linux, and macOS 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 SSL key generation.

Behavior

If set to *string*, the specified entropy-source file or device seeds the random number generator used for 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 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 SSL key generation.

`/dev/hwrng`

is a hardware random number generator. The behavior is dependent on the device used in your environment.

Notes

- This option is ignored when SSL is disabled (`EncryptionMethod=0`) or the seed source is set to Poll Only (`PRNGSeedSource=1`).
- For processes that employ multiple 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 243

PRNGSeedSource



Supported on UNIX, Linux, and macOS only.

Attribute

PRNGSeedSource (PSS)

Purpose

Specifies the source of the seed the driver uses for SSL key generation. Seeds are a pseudorandom or random value used to set the initial state of the random number generator used to generate SSL keys. Using seeds with a higher level of entropy, or randomness, provides a more secure transmission of data encrypted using 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 SSL key generation.

If set to 1 (Poll Only) , the driver uses the RAND_poll function in SSL to create the seed used for SSL key generation.

Notes

- For processes that employ multiple 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 SSL is disabled (`EncryptionMethod=0`)

Default

0 (File)

GUI Tab

NA

See also

[PRNGSeedFile](#) on page 242

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 115

See also

[Support of Materialized Views](#) on page 54

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 115

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 10g 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 12c 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 115

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 115

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](#)

SSLibName

Attribute

SSLibName (SLN)

Purpose

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

This option allows you to designate a different 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 SSL Library to be used by your data source or connection.

Example

```
C:\Program Files\Progress\DataDirect\ODBC_80\Drivers\OpenSSL\1.0.2d\ddssl28.dll
```

Notes

- The OpenSSL library files provided by Progress combine the cryptographic and SSL libraries into a single file; therefore, when your drivers are using a Progress library file, the values specified for the CryptoLibName and SSLibName 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.

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:

```
SSLibName=C:\Program Files\Progress\DataDirect\  
ODBC_80\Drivers\OpenSSL\1.0.2r\ddssl28.dll;
```

See also

- [CryptoLibName](#) on page 204
- [Advanced Tab](#) on page 80

Support Binary XML

Attribute

SupportBinaryXML (SBX)

Purpose

Enables the driver to support XMLType with binary storage on servers running Oracle 12c 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

- 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 199
- [Connection Retry Delay](#) on page 200

Trust Store

Attribute

Truststore (TS)

Purpose

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

Valid Values

truststore_directory\filename

where:

truststore_directory

is the directory where the truststore file is located

filename

is the file name of the truststore file.

Notes

- The truststore and keystore files may be the same file.

Default

None

GUI Tab

[Security tab](#)

Trust Store Password

TruststorePassword (TSP)

Purpose

Specifies the password that is used to access the truststore file when 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 115

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

Valid Values

0 | 1

Behavior

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

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

Notes

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

Default

1 (Enabled)

GUI Tab

[Security tab](#)

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=14` (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 136

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 115

Reference

This part provides detailed reference information about your Progress DataDirect *for* ODBC driver.

For details, see the following topics:

- [Code Page Values](#)
- [ODBC API and Scalar Functions](#)
- [Internationalization, Localization, and Unicode](#)
- [Designing ODBC Applications for Performance Optimization](#)
- [Using Indexes](#)
- [Locking and Isolation Levels](#)
- [SSL Encryption Cipher Suites](#)
- [DataDirect Bulk Load](#)
- [Threading](#)
- [WorkAround Options](#)

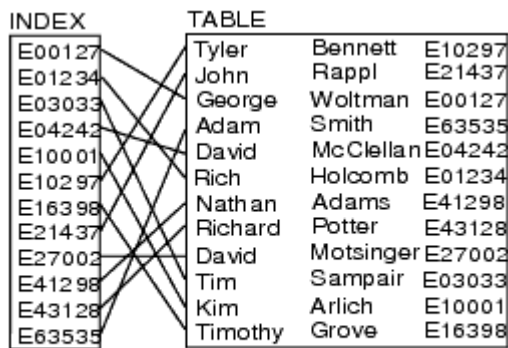
Using Indexes

This chapter discusses the ways in which you can improve the performance of database activity using indexes. It provides general guidelines that apply to most databases. Consult your database vendor's documentation for more detailed information.

Introduction

An index is a database structure that you can use to improve the performance of database activity. A database table can have one or more indexes associated with it.

An index is defined by a field expression that you specify when you create the index. Typically, the field expression is a single field name, like `emp_id`. An index created on the `emp_id` field, for example, contains a sorted list of the employee ID values in the table. Each value in the list is accompanied by references to the rows that contain that value.



A database driver can use indexes to find rows quickly. An index on the `emp_id` field, for example, greatly reduces the time that the driver spends searching for a particular employee ID value. Consider the following Where clause:

```
WHERE EMP_id = 'E10001'
```

Without an index, the server must search the entire database table to find those rows having an employee ID of E10001. By using an index on the `emp_id` field, however, the server can quickly find those rows.

Indexes may improve the performance of SQL statements. You may not notice this improvement with small tables, but it can be significant for large tables; however, there can be disadvantages to having too many indexes. Indexes can slow down the performance of some inserts, updates, and deletes when the driver has to maintain the indexes as well as the database tables. Also, indexes take additional disk space.

Improving Row Selection Performance

For indexes to improve the performance of selections, the index expression must match the selection condition exactly. For example, if you have created an index whose expression is `last_name`, the following Select statement uses the index:

```
SELECT * FROM emp WHERE last_name = 'Smith'
```

This Select statement, however, does not use the index:

```
SELECT * FROM emp WHERE UPPER(last_name) = 'SMITH'
```

The second statement does not use the index because the Where clause contains `UPPER(last_name)`, which does not match the index expression `last_name`. If you plan to use the `UPPER` function in all your Select statements and your database supports indexes on expressions, then you should define an index using the expression `UPPER(last_name)`.

Indexing Multiple Fields

If you often use Where clauses that involve more than one field, you may want to build an index containing multiple fields. Consider the following Where clause:

```
WHERE last_name = 'Smith' AND first_name = 'Thomas'
```

For this condition, the optimal index field expression is last_name, first_name. This creates a concatenated index.

Concatenated indexes can also be used for Where clauses that contain only the first of two concatenated fields. The last_name, first_name index also improves the performance of the following Where clause (even though no first name value is specified):

```
last_name = 'Smith'
```

Consider the following Where clause:

```
WHERE last_name = 'Smith' AND middle_name = 'Edward' AND first_name = 'Thomas'
```

If your index fields include all the conditions of the Where clause in that order, the driver can use the entire index. If, however, your index is on two nonconsecutive fields, for example, last_name and first_name, the driver can use only the last_name field of the index.

The driver uses only one index when processing Where clauses. If you have complex Where clauses that involve a number of conditions for different fields and have indexes on more than one field, the driver chooses an index to use. The driver attempts to use indexes on conditions that use the equal sign as the relational operator rather than conditions using other operators (such as greater than). Assume you have an index on the emp_id field as well as the last_name field and the following Where clause:

```
WHERE emp_id >= 'E10001' AND last_name = 'Smith'
```

In this case, the driver selects the index on the last_name field.

If no conditions have the equal sign, the driver first attempts to use an index on a condition that has a lower *and* upper bound, and then attempts to use an index on a condition that has a lower *or* upper bound. The driver always attempts to use the most restrictive index that satisfies the Where clause.

In most cases, the driver does not use an index if the Where clause contains an OR comparison operator. For example, the driver does not use an index for the following Where clause:

```
WHERE emp_id >= 'E10001' OR last_name = 'Smith'
```

Deciding Which Indexes to Create

Before you create indexes for a database table, consider how you will use the table. The most common operations on a table are:

- Inserting, updating, and deleting rows
- Retrieving rows

If you most often insert, update, and delete rows, then the fewer indexes associated with the table, the better the performance. This is because the driver must maintain the indexes as well as the database tables, thus slowing down the performance of row inserts, updates, and deletes. It may be more efficient to drop all indexes before modifying a large number of rows, and re-create the indexes after the modifications.

If you most often retrieve rows, you must look further to define the criteria for retrieving rows and create indexes to improve the performance of these retrievals. Assume you have an employee database table and you will retrieve rows based on employee name, department, or hire date. You would create three indexes—one on the dept field, one on the hire_date field, and one on the last_name field. Or perhaps, for the retrievals based on the name field, you would want an index that concatenates the last_name and the first_name fields (see "Indexing Multiple Fields" for details).

Here are a few rules to help you decide which indexes to create:

- If your row retrievals are based on only one field at a time (for example, dept = 'D101'), create an index on these fields.
- If your row retrievals are based on a combination of fields, look at the combinations.
- If the comparison operator for the conditions is And (for example, city = 'Raleigh' AND state = 'NC'), then build a concatenated index on the city and state fields. This index is also useful for retrieving rows based on the city field.
- If the comparison operator is OR (for example, dept = 'D101' OR hire_date > {01/30/89}), an index does not help performance. Therefore, you need not create one.
- If the retrieval conditions contain both AND and OR comparison operators, you can use an index if the OR conditions are grouped. For example:

```
dept = 'D101' AND (hire_date > {01/30/89} OR exempt = 1)
```

In this case, an index on the dept field improves performance.

- If the AND conditions are grouped, an index does not improve performance. For example:

```
(dept = 'D101' AND hire_date) > {01/30/89} OR exempt = 1
```

See also

[Indexing Multiple Fields](#) on page 265

Improving Join Performance

When joining database tables, index tables can greatly improve performance. Unless the proper indexes are available, queries that use joins can take a long time.

Assume you have the following Select statement:

```
SELECT * FROM dept, emp WHERE dept.dept_id = emp.dept_id
```

In this example, the dept and emp database tables are being joined using the dept_id field. When the driver executes a query that contains a join, it processes the tables from left to right and uses an index on the second table's join field (the dept field of the emp table). To improve join performance, you need an index on the join field of the second table in the FROM clause.

If the FROM clause includes a third table, the driver also uses an index on the field in the third table that joins it to any previous table. For example:

```
SELECT * FROM dept, emp, addr WHERE dept.dept_id = emp.dept AND emp.loc = addr.loc
```

In this case, you should have an index on the emp.dept field and the addr.loc field.

Code Page Values

This chapter lists supported code page values, along with a description, for your driver.

For details, see the following topics:

- [IANAAppCodePage Values](#)

IANAAppCodePage Values

The following table lists supported code page values for the IANAAppCodePage connection option. See "IANAAppCodePage" for information about this attribute.

To determine the correct numeric value (the MIBenum value) for the IANAAppCodePage connection string attribute, perform the following steps:

1. Determine the code page of your database.
2. Determine the MIBenum value that corresponds to your database code page. To do this, go to:

<http://www.iana.org/assignments/character-sets>

On this web page, search for the name of your database code page. This name will be listed as an alias or the name of a character set, and will have a MIBenum value associated with it.

3. Check the following table to make sure that the MIBenum value you looked up on the IANA Web page is supported by your Progress DataDirect *for* ODBC driver. If the value is not listed, contact Progress Technical Support to request support for that value.

Table 12: IANAAppCodePage Values

Value (MIBenum)	Description
3	US_ASCII
4	ISO_8859_1
5	ISO_8859_2
6	ISO_8859_3
7	ISO_8859_4
8	ISO_8859_5
9	ISO_8859_6
10	ISO_8859_7
11	ISO_8859_8
12	ISO_8859_9
16	JIS_Encoding
17	Shift_JIS
18	EUC_JP
30	ISO_646_IRV
36	KS_C_5601
37	ISO_2022_KR
38	EUC_KR
39	ISO_2022_JP
40	ISO_2022_JP_2
57	GB_2312_80
104	ISO_2022_CN
105	ISO_2022_CN_EXT
109	ISO_8859_13
110	ISO_8859_14
111	ISO_8859_15

Value (MIBenum)	Description
113	GBK
2004	HP_ROMAN8
2009	IBM850
2010	IBM852
2011	IBM437
2013	IBM862
2016	IBM-Thai
2024	WINDOWS-31J
2025	GB2312
2026	Big5
2027	MACINTOSH
2028	IBM037
2029	IBM038
2030	IBM273
2033	IBM277
2034	IBM278
2035	IBM280
2037	IBM284
2038	IBM285
2039	IBM290
2040	IBM297
2041	IBM420
2043	IBM424
2044	IBM500
2045	IBM851
2046	IBM855

Value (MIBenum)	Description
2047	IBM857
2048	IBM860
2049	IBM861
2050	IBM863
2051	IBM864
2052	IBM865
2053	IBM868
2054	IBM869
2055	IBM870
2056	IBM871
2062	IBM918
2063	IBM1026
2084	KOI8_R
2085	HZ_GB_2312
2086	IBM866
2087	IBM775
2089	IBM00858
2091	IBM01140
2092	IBM01141
2093	IBM01142
2094	IBM01143
2095	IBM01144
2096	IBM01145
2097	IBM01146
2098	IBM01147
2099	IBM01148

Value (MIBenum)	Description
2100	IBM01149
2102	IBM1047
2250	WINDOWS_1250
2251	WINDOWS_1251
2252	WINDOWS_1252
2253	WINDOWS_1253
2254	WINDOWS_1254
2255	WINDOWS_1255
2256	WINDOWS_1256
2257	WINDOWS_1257
2258	WINDOWS_1258
2259	TIS_620
200000939 ⁸	IBM-939
200000943 ⁸	IBM-943_P14A-2000
200001025 ⁸	IBM-1025
200004396 ⁸	IBM-4396
200005026 ⁸	IBM-5026
200005035 ⁸	IBM-5035

See also

[IANAAppCodePage](#) on page 224

[Contacting Technical Support](#) on page 20

⁸ These values are assigned by Progress DataDirect and do not appear in <http://www.iana.org/assignments/character-sets>.

ODBC API and Scalar Functions

This chapter lists the ODBC API functions that your Progress DataDirect *for* ODBC driver supports. In addition, it lists the scalar functions that you use in SQL statements.

For details, see the following topics:

- [API Functions](#)
- [Scalar Functions](#)

API Functions

Your Progress DataDirect *for* ODBC driver is Level 1 compliant, that is, it supports all ODBC Core and Level 1 functions. It also supports a limited set of Level 2 functions, as described in the following table.

Table 13: Function Conformance for ODBC 2.x Applications

Core Functions	Level 1 Functions	Level 2 Functions
SQLAllocConnect ⁹	SQLColumns	SQLBrowseConnect
SQLAllocEnv ⁹	SQLDriverConnect	SQLDataSources
SQLAllocStmt ⁹	SQLGetConnectOption	SQLDescribeParam ⁹
SQLBindCol	SQLGetData	SQLExtendedFetch (forward scrolling only)
SQLBindParameter	SQLGetFunctions	SQLMoreResults
SQLCancel	SQLGetInfo	SQLNativeSql
SQLColAttributes	SQLGetStmtOption ⁹	SQLNumParams
SQLConnect	SQLGetTypeInfo	SQLParamOptions ⁹
SQLDescribeCol	SQLParamData	SQLSetScrollOptions
SQLDisconnect	SQLPutData	
SQLDrivers	SQLSetConnectOption	
SQLError	SQLSetStmtOption ⁹	
SQLExecDirect	SQLSpecialColumns	
SQLExecute	SQLStatistics	
SQLFetch	SQLTables	
SQLFreeConnect ⁹		
SQLFreeEnv ⁹		
SQLFreeStmt		
SQLGetCursorName		
SQLNumResultCols		
SQLPrepare		
SQLRowCount		
SQLSetCursorName		
SQLTransact ⁹		

The functions for ODBC 3.x Applications that the driver supports are listed in the following table. Any additions to these supported functions or differences in the support of specific functions are listed in "ODBC Compliance".

⁹ For macOS, this function is not supported by the iODBC driver manager; therefore, it cannot currently be executed by the driver.

Table 14: Function Conformance for ODBC 3.x Applications

SQLAllocHandle	SQLGetDescField
SQLBindCol	SQLGetDescRec
SQLBindParameter	SQLGetDiagField
SQLBrowseConnect (except for Progress)	SQLGetDiagRec
SQLBulkOperations	SQLGetEnvAttr
SQLCancel	SQLGetFunctions
SQLCloseCursor	SQLGetInfo
SQLColAttribute	SQLGetStmtAttr
SQLColumns	SQLGetTypeInfo
SQLConnect	SQLMoreResults
SQLCopyDesc	SQLNativeSql
SQLDataSources	SQLNumParens
SQLDescribeCol	SQLNumResultCols
SQLDisconnect	SQLParamData
SQLDriverConnect	SQLPrepare
SQLDrivers	SQLPutData
SQLEndTran	SQLRowCount
SQLError	SQLSetConnectAttr
SQLExecDirect	SQLSetCursorName
SQLExecute	SQLSetDescField
SQLExtendedFetch	SQLSetDescRec
SQLFetch	SQLSetEnvAttr
SQLFetchScroll (forward scrolling only)	SQLSetStmtAttr
SQLFreeHandle	SQLSpecialColumns
SQLFreeStmt	SQLStatistics
SQLGetConnectAttr	SQLTables
SQLGetCursorName	SQLTransact
SQLGetData	

See also

[ODBC Compliance](#) on page 34

Scalar Functions

This section lists the scalar functions that ODBC supports. Your database system may not support all these functions. Refer to the documentation for your database system to find out which functions are supported. Also, depending on the driver that you are using, all the scalar functions may not be supported. To check which scalar functions are supported by a driver, use the SQLGetInfo ODBC function.

You can use these scalar functions in SQL statements using the following syntax:

```
{fn scalar-function}
```

where *scalar-function* is one of the functions listed in the following tables. For example:

```
SELECT {fn UCASE(NAME)} FROM EMP
```

Table 15: Scalar Functions

String Functions	Numeric Functions	Timedate Functions	System Functions
ASCII	ABS	CURDATE	CURSESSIONID
BIT_LENGTH	ACOS	CURTIME	CURRENT_USER
CHAR	ASIN	DATEDIFF	DATABASE
CHAR_LENGTH	ATAN	DAYNAME	IDENTITY
CONCAT	ATAN2	DAYOFMONTH	USER
DIFFERENCE	BITAND	DAYOFWEEK	
HEXTORAW	BITOR	DAYOFYEAR	
INSERT	CEILING	EXTRACT	
LCASE	COS	HOUR	
LEFT	COT	MINUTE	
LENGTH	DEGREES	MONTH	
LOCATE	EXP	MONTHNAME	
LOWER	FLOOR	NOW	
LTRIM	LOG	QUARTER	
OCTET_LENGTH	LOG10	SECOND	
RAWTOHEX	MOD	WEEK	
REPEAT	PI	YEAR	
REPLACE	POWER	CURRENT_DATE	

String Functions	Numeric Functions	Timedate Functions	System Functions
RIGHT	RADIANS	CURRENT_TIME	
RTRIM	RAND	CURRENT_TIMESTAMP	
SOUNDEX	ROUND		
SPACE	ROUNDMAGIC		
SUBSTR	SIGN		
SUBSTRING	SIN		
UCASE	SQRT		
UPPER	TAN		
	TRUNCATE		

String Functions

The following table lists the string functions that ODBC supports.

The string functions listed accept the following arguments:

- *string_exp* can be the name of a column, a string literal, or the result of another scalar function, where the underlying data type is SQL_CHAR, SQL_VARCHAR, or SQL_LONGVARCHAR.
- *start*, *length*, and *count* can be the result of another scalar function or a literal numeric value, where the underlying data type is SQL_TINYINT, SQL_SMALLINT, or SQL_INTEGER.

The string functions are one-based; that is, the first character in the string is character 1.

Character string literals must be surrounded in single quotation marks.

Table 16: Scalar String Functions

Function	Returns
ASCII(<i>string_exp</i>)	ASCII code value of the leftmost character of <i>string_exp</i> as an integer.
BIT_LENGTH(<i>string_exp</i>) [ODBC 3.0 only]	The length in bits of the string expression.
CHAR(<i>code</i>)	The character with the ASCII code value specified by <i>code</i> . <i>code</i> should be between 0 and 255; otherwise, the return value is data-source dependent.
CHAR_LENGTH(<i>string_exp</i>) [ODBC 3.0 only]	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHARACTER_LENGTH function.)

Function	Returns
CHARACTER_LENGTH(<i>string_exp</i>) [ODBC 3.0 only]	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHAR_LENGTH function.)
CONCAT(<i>string_exp1</i> , <i>string_exp2</i>)	The string resulting from concatenating <i>string_exp2</i> and <i>string_exp1</i> . The string is system dependent.
DIFFERENCE(<i>string_exp1</i> , <i>string_exp2</i>)	An integer value that indicates the difference between the values returned by the SOUNDEX function for <i>string_exp1</i> and <i>string_exp2</i> .
INSERT(<i>string_exp1</i> , <i>start</i> , <i>length</i> , <i>string_exp2</i>)	A string where <i>length</i> characters have been deleted from <i>string_exp1</i> beginning at <i>start</i> and where <i>string_exp2</i> has been inserted into <i>string_exp</i> beginning at <i>start</i> .
LCASE(<i>string_exp</i>)	Uppercase characters in <i>string_exp</i> converted to lowercase.
LEFT(<i>string_exp</i> , <i>count</i>)	The <i>count</i> of characters of <i>string_exp</i> .
LENGTH(<i>string_exp</i>)	The number of characters in <i>string_exp</i> , excluding trailing blanks and the string termination character.
LOCATE(<i>string_exp1</i> , <i>string_exp2</i> [, <i>start</i>])	The starting position of the first occurrence of <i>string_exp1</i> within <i>string_exp2</i> . If <i>start</i> is not specified, the search begins with the first character position in <i>string_exp2</i> . If <i>start</i> is specified, the search begins with the character position indicated by the value of <i>start</i> . The first character position in <i>string_exp2</i> is indicated by the value 1. If <i>string_exp1</i> is not found, 0 is returned.
LTRIM(<i>string_exp</i>)	The characters of <i>string_exp</i> with leading blanks removed.
OCTET_LENGTH(<i>string_exp</i>) [ODBC 3.0 only]	The length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8.
POSITION(<i>character_exp</i> IN <i>character_exp</i>) [ODBC 3.0 only]	The position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0.
REPEAT(<i>string_exp</i> , <i>count</i>)	A string composed of <i>string_exp</i> repeated <i>count</i> times.
REPLACE(<i>string_exp1</i> , <i>string_exp2</i> , <i>string_exp3</i>)	Replaces all occurrences of <i>string_exp2</i> in <i>string_exp1</i> with <i>string_exp3</i> .
RIGHT(<i>string_exp</i> , <i>count</i>)	The rightmost <i>count</i> of characters in <i>string_exp</i> .
RTRIM(<i>string_exp</i>)	The characters of <i>string_exp</i> with trailing blanks removed.
SOUNDEX(<i>string_exp</i>)	A data source dependent string representing the sound of the words in <i>string_exp</i> .

Function	Returns
SPACE(<i>count</i>)	A string consisting of <i>count</i> spaces.
SUBSTRING(<i>string_exp</i> , <i>start</i> , <i>length</i>)	A string derived from <i>string_exp</i> beginning at the character position <i>start</i> for <i>length</i> characters.
UCASE(<i>string_exp</i>)	Lowercase characters in <i>string_exp</i> converted to uppercase.

Numeric Functions

The following table lists the numeric functions that ODBC supports.

The numeric functions listed accept the following arguments:

- *numeric_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL_NUMERIC, SQL_DECIMAL, SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, SQL_BIGINT, SQL_FLOAT, SQL_REAL, or SQL_DOUBLE.
- *float_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL_FLOAT.
- *integer_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL_TINYINT, SQL_SMALLINT, SQL_INTEGER, or SQL_BIGINT.

Table 17: Scalar Numeric Functions

Function	Returns
ABS(<i>numeric_exp</i>)	Absolute value of <i>numeric_exp</i> .
ACOS(<i>float_exp</i>)	Arccosine of <i>float_exp</i> as an angle in radians.
ASIN(<i>float_exp</i>)	Arcsine of <i>float_exp</i> as an angle in radians.
ATAN(<i>float_exp</i>)	Arctangent of <i>float_exp</i> as an angle in radians.
ATAN2(<i>float_exp1</i> , <i>float_exp2</i>)	Arctangent of the x and y coordinates, specified by <i>float_exp1</i> and <i>float_exp2</i> as an angle in radians.
CEILING(<i>numeric_exp</i>)	Smallest integer greater than or equal to <i>numeric_exp</i> .
COS(<i>float_exp</i>)	Cosine of <i>float_exp</i> as an angle in radians.
COT(<i>float_exp</i>)	Cotangent of <i>float_exp</i> as an angle in radians.
DEGREES(<i>numeric_exp</i>)	Number if degrees converted from <i>numeric_exp</i> radians.
EXP(<i>float_exp</i>)	Exponential value of <i>float_exp</i> .
FLOOR(<i>numeric_exp</i>)	Largest integer less than or equal to <i>numeric_exp</i> .
LOG(<i>float_exp</i>)	Natural log of <i>float_exp</i> .

Function	Returns
LOG10(<i>float_exp</i>)	Base 10 log of <i>float_exp</i> .
MOD(<i>integer_exp1</i> , <i>integer_exp2</i>)	Remainder of <i>integer_exp1</i> divided by <i>integer_exp2</i> .
PI()	Constant value of pi as a floating-point number.
POWER(<i>numeric_exp</i> , <i>integer_exp</i>)	Value of <i>numeric_exp</i> to the power of <i>integer_exp</i> .
RADIANS(<i>numeric_exp</i>)	Number of radians converted from <i>numeric_exp</i> degrees.
RAND([<i>integer_exp</i>])	Random floating-point value using <i>integer_exp</i> as the optional seed value.
ROUND(<i>numeric_exp</i> , <i>integer_exp</i>)	<i>numeric_exp</i> rounded to <i>integer_exp</i> places right of the decimal (left of the decimal if <i>integer_exp</i> is negative).
SIGN(<i>numeric_exp</i>)	Indicator of the sign of <i>numeric_exp</i> . If <i>numeric_exp</i> < 0, -1 is returned. If <i>numeric_exp</i> = 0, 0 is returned. If <i>numeric_exp</i> > 0, 1 is returned.
SIN(<i>float_exp</i>)	Sine of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
SQRT(<i>float_exp</i>)	Square root of <i>float_exp</i> .
TAN(<i>float_exp</i>)	Tangent of <i>float_exp</i> , where <i>float_exp</i> is an angle in radians.
TRUNCATE(<i>numeric_exp</i> , <i>integer_exp</i>)	<i>numeric_exp</i> truncated to <i>integer_exp</i> places right of the decimal. (If <i>integer_exp</i> is negative, truncation is to the left of the decimal.)

Date and Time Functions

The following table lists the date and time functions that ODBC supports.

The date and time functions listed accept the following arguments:

- *date_exp* can be a column name, a date or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL_CHAR, SQL_VARCHAR, SQL_DATE, or SQL_TIMESTAMP.
- *time_exp* can be a column name, a timestamp or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL_CHAR, SQL_VARCHAR, SQL_TIME, or SQL_TIMESTAMP.
- *timestamp_exp* can be a column name; a time, date, or timestamp literal; or the result of another scalar function, where the underlying data type can be represented as SQL_CHAR, SQL_VARCHAR, SQL_TIME, SQL_DATE, or SQL_TIMESTAMP.

Table 18: Scalar Time and Date Functions

Function	Returns
CURRENT_DATE() [ODBC 3.0 only]	Current date.

Function	Returns
CURRENT_TIME(<i>time-precision</i>) [ODBC 3.0 only]	Current local time. The <i>time-precision</i> argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP(<i>timestamp-precision</i>) [ODBC 3.0 only]	Current local date and local time as a timestamp value. The <i>timestamp-precision</i> argument determines the seconds precision of the returned timestamp.
CURDATE()	Current date as a date value.
CURTIME()	Current local time as a time value.
DAYNAME(<i>date_exp</i>)	Character string containing a data-source-specific name of the day for the day portion of <i>date_exp</i> .
DAYOFMONTH(<i>date_exp</i>)	Day of the month in <i>date_exp</i> as an integer value (1–31).
DAYOFWEEK(<i>date_exp</i>)	Day of the week in <i>date_exp</i> as an integer value (1–7).
DAYOFYEAR(<i>date_exp</i>)	Day of the year in <i>date_exp</i> as an integer value (1–366).
EXTRACT({YEAR MONTH DAY HOUR MINUTE SECOND} FROM <i>datetime_value</i>)	Any of the date and time terms can be extracted from <i>datetime_value</i> .
HOUR(<i>time_exp</i>)	Hour in <i>time_exp</i> as an integer value (0–23).
MINUTE(<i>time_exp</i>)	Minute in <i>time_exp</i> as an integer value (0–59).
MONTH(<i>date_exp</i>)	Month in <i>date_exp</i> as an integer value (1–12).
MONTHNAME(<i>date_exp</i>)	Character string containing the data source-specific name of the month.
NOW()	Current date and time as a timestamp value.
QUARTER(<i>date_exp</i>)	Quarter in <i>date_exp</i> as an integer value (1–4).
SECOND(<i>time_exp</i>)	Second in <i>date_exp</i> as an integer value (0–59).

Function	Returns
TIMESTAMPADD(<i>interval</i> , <i>integer_exp</i> , <i>time_exp</i>)	Timestamp calculated by adding <i>integer_exp</i> intervals of type <i>interval</i> to <i>time_exp</i> . <i>interval</i> can be one of the following values: SQL_TSI_FRAC_SECOND SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR Fractional seconds are expressed in billionths of a second.
TIMESTAMPDIFF(<i>interval</i> , <i>time_exp1</i> , <i>time_exp2</i>)	Integer number of intervals of type <i>interval</i> by which <i>time_exp2</i> is greater than <i>time_exp1</i> . <i>interval</i> has the same values as TIMESTAMPADD. Fractional seconds are expressed in billionths of a second.
WEEK(<i>date_exp</i>)	Week of the year in <i>date_exp</i> as an integer value (1–53).
YEAR(<i>date_exp</i>)	Year in <i>date_exp</i> . The range is data-source dependent.

System Functions

The following table lists the system functions that ODBC supports.

Table 19: Scalar System Functions

Function	Returns
DATABASE()	Name of the database, corresponding to the connection handle (<i>hdbc</i>).
IFNULL(<i>exp</i> , <i>value</i>)	<i>value</i> , if <i>exp</i> is null.
USER()	Authorization name of the user.

Internationalization, Localization, and Unicode

This chapter provides an overview of how internationalization, localization, and Unicode relate to each other. It also provides a background on Unicode, and how it is accommodated by Unicode and non-Unicode ODBC drivers.

For details, see the following topics:

- [Internationalization and Localization](#)
- [Unicode Character Encoding](#)
- [Unicode and Non-Unicode ODBC Drivers](#)
- [Driver Manager and Unicode Encoding on UNIX/Linux](#)
- [Character Encoding in the `odbc.ini` and `odbcinst.ini` Files](#)

Internationalization and Localization

Software that has been designed for *internationalization* is able to manage different linguistic and cultural conventions transparently and without modification. The same binary copy of an application should run on any localized version of an operating system without requiring source code changes.

Software that has been designed for *localization* includes language translation (such as text messages, icons, and buttons), cultural data (such as dates, times, and currency), and other components (such as input methods and spell checkers) for meeting regional market requirements.

Properly designed applications can accommodate a localized interface without extensive modification. The applications can be designed, first, to run internationally, and, second, to accommodate the language- and cultural-specific elements of a designated locale.

Locale

A locale represents the language and cultural data chosen by the user and dynamically loaded into memory at runtime. The locale settings are applied to the operating system and to subsequent application launches.

While language is a fairly straightforward item, cultural data is a little more complex. Dates, numbers, and currency are all examples of data that is formatted according to cultural expectations. Because cultural preferences are bound to a geographic area, country is an important element of locale. Together these two elements (language and country) provide a precise context in which information can be presented. Locale presents information in the language and form that is best understood and appreciated by the local user.

Language

A locale's language is specified by the ISO 639 standard. The following table lists some commonly used language codes.

Language Code	Language
en	English
nl	Dutch
fr	French
es	Spanish
zh	Chinese
ja	Japanese
vi	Vietnamese

Because language is correlated with geography, a language code might not capture all the nuances of usage in a particular area. For example, French and Canadian French may use different phrases and terms to mean different things even though basic grammar and vocabulary are the same. Language is only one element of locale.

Country

The locale's country identifier is also specified by an ISO standard, ISO 3166, which describes valid two-letter codes for all countries. ISO 3166 defines these codes in uppercase letters. The following table lists some commonly used country codes.

Country Code	Country
US	United States
FR	France
IE	Ireland
CA	Canada
MX	Mexico

The country code provides more contextual information for a locale and affects a language's usage, word spelling, and collation rules.

Variant

A variant is an optional extension to a locale. It identifies a custom locale that is not possible to create with just language and country codes. Variants can be used by anyone to add additional context for identifying a locale. The locale `en_US` represents English (United States), but `en_US_CA` represents even more information and might identify a locale for English (California, U.S.A). Operating system or software vendors can use these variants to create more descriptive locales for their specific environments.

Unicode Character Encoding

In addition to locale, the other major component of internationalizing software is the use of the Universal Codeset, or Unicode. Most developers know that Unicode is a standard encoding that can be used to support multilingual character sets. Unfortunately, understanding Unicode is not as simple as its name would indicate. Software developers have used a number of character encodings, from ASCII to Unicode, to solve the many problems that arise when developing software applications that can be used worldwide.

Background

Most legacy computing environments have used ASCII character encoding developed by the ANSI standards body to store and manipulate character strings inside software applications. ASCII encoding was convenient for programmers because each ASCII character could be stored as a byte. The initial version of ASCII used only 7 of the 8 bits available in a byte, which meant that applications could use only 128 different characters. This version of ASCII could not account for European characters and was completely inadequate for Asian characters. Using the eighth bit to extend the total range of characters to 256 added support for most European characters. Today, ASCII refers to either the 7-bit or 8-bit encoding of characters.

As the need increased for applications with additional international support, ANSI again increased the functionality of ASCII by developing an extension to accommodate multilingual software. The extension, known as the Double-Byte Character Set (DBCS), allowed existing applications to function without change, but provided for the use of additional characters, including complex Asian characters. With DBCS, characters map to either one byte (for example, American ASCII characters) or two bytes (for example, Asian characters). The DBCS environment also introduced the concept of an operating system code page that identified how characters would be encoded into byte sequences in a particular computing environment. DBCS encoding provided a cross-platform mechanism for building multilingual applications.

The DataDirect *for* ODBC UNIX, Linux, and macOS drivers can use double-byte character sets. The drivers normally use the character set defined by the default locale "C" unless explicitly pointed to another character set. The default locale "C" corresponds to the 7-bit US-ASCII character set. Use the following procedure to set the locale to a different character set:

1. Add the following line at the beginning of applications that use double-byte character sets:

```
setlocale (LC_ALL, "");
```

This is a standard function for UNIX-based platforms. It selects the character set indicated by the environment variable `LANG` as the one to be used by X/Open compliant, character-handling functions. If this line is not present, or if `LANG` is not set or is set to `NULL`, the default locale "C" is used.

2. Set the `LANG` environment variable to the appropriate character set. The command `locale -a` can be used to display all supported character sets on your system.

For more information, refer to the man pages for "locale" and "setlocale."

Using a DBCS, however, was not ideal; many developers felt that there was a better way to solve the problem. A group of leading software companies joined forces to form the Unicode Consortium. Together, they produced a new solution to building worldwide applications—Unicode. Unicode was originally designed as a fixed-width, uniform two-byte designation that could represent all modern scripts without the use of code pages. The Unicode Consortium has continued to evaluate new characters, and the current number of supported characters is over 109,000.

Although it seemed to be the perfect solution to building multilingual applications, Unicode started off with a significant drawback—it would have to be retrofitted into existing computing environments. To use the new paradigm, all applications would have to change. As a result, several standards-based transliterations were designed to convert two-byte fixed Unicode values into more appropriate character encodings, including, among others, UTF-8, UCS-2, UTF-16, and UTF-32.

UTF-8 is a standard method for transforming Unicode values into byte sequences that maintain transparency for all ASCII codes. UTF-8 is recognized by the Unicode Consortium as a mechanism for transforming Unicode values and is popular for use with HTML, XML, and other protocols. UTF-8 is, however, currently used primarily on AIX, HP-UX, Solaris, and Linux.

UCS-2 encoding is a fixed, two-byte encoding sequence and is a method for transforming Unicode values into byte sequences. It is the standard for Windows 95, Windows 98, Windows Me, and Windows NT.

UTF-16 is a superset of UCS-2, with the addition of some special characters in surrogate pairs. UTF-16 is the standard encoding for Windows 7 and higher. Microsoft recommends using UTF-16 for new applications.

UTF-32 encoding is a fixed-width, 4 byte method for transforming Unicode values into byte sequences. It is capable of defining all Unicode characters and is common for macOS platforms.

See "Unicode Support" to determine which encoding formats your driver supports.

See also

[Unicode Support](#) on page 51

Unicode Support in Databases

Recently, database vendors have begun to support Unicode data types natively in their systems. With Unicode support, one database can hold multiple languages. For example, a large multinational corporation could store expense data in the local languages for the Japanese, U.S., English, German, and French offices in one database.

Not surprisingly, the implementation of Unicode data types varies from vendor to vendor. For example, the Microsoft SQL Server 2000 implementation of Unicode provides data in UTF-16 format, while Oracle provides Unicode data types in UTF-8 and UTF-16 formats. A consistent implementation of Unicode not only depends on the operating system, but also on the database itself.

Unicode Support in ODBC

Prior to the ODBC 3.5 standard, all ODBC access to function calls and string data types was through ANSI encoding (either ASCII or DBCS). Applications and drivers were both ANSI-based.

The ODBC 3.5 standard specified that the ODBC Driver Manager be capable of mapping both Unicode function calls and string data types to ANSI encoding as transparently as possible. This meant that ODBC 3.5-compliant Unicode applications could use Unicode function calls and string data types with ANSI drivers because the Driver Manager could convert them to ANSI. Because of character limitations in ANSI, however, not all conversions are possible.

The ODBC Driver Manager version 3.5 and later, therefore, supports the following configurations:

- ANSI application with an ANSI driver

-
- ANSI application with a Unicode driver
 - Unicode application with a Unicode driver
 - Unicode application with an ANSI driver

A Unicode application can work with an ANSI driver because the Driver Manager provides limited Unicode-to-ANSI mapping. The Driver Manager makes it possible for a pre-3.5 ANSI driver to work with a Unicode application. What distinguishes a Unicode driver from a non-Unicode driver is the Unicode driver's capacity to interpret Unicode function calls without the intervention of the Driver Manager, as described in the following section.

Unicode and Non-Unicode ODBC Drivers

The way in which a driver handles function calls from a Unicode application determines whether it is considered a "Unicode driver."

Function Calls

Instead of the standard ANSI SQL function calls, such as `SQLConnect`, Unicode applications use "W" (wide) function calls, such as `SQLConnectW`. If the driver is a true Unicode driver, it can understand "W" function calls and the Driver Manager can pass them through to the driver without conversion to ANSI. The Progress DataDirect *for* ODBC for Oracle Wire Protocol Driver supports "W" function calls.

If a driver is a non-Unicode driver, it cannot understand W function calls, and the Driver Manager must convert them to ANSI calls before sending them to the driver. The Driver Manager determines the ANSI encoding system to which it must convert by referring to a code page. On Windows, this reference is to the Active Code Page. On non-Windows platforms, it is to the `IANAAppCodePage` connection string attribute, part of the `odbc.ini` file.

The following examples illustrate these conversion streams for the Progress DataDirect for ODBC drivers. The Driver Manager on UNIX and Linux determines the type of Unicode encoding of both the application and the driver, and performs conversions when the application and driver use different types of encoding. This determination is made by checking two ODBC attributes: `SQL_ATTR_APP_UNICODE_TYPE` and `SQL_ATTR_DRIVER_UNICODE_TYPE`, which can be set for either the environment, using `SQLSetEnvAttr`, or the connection, using `SQLSetConnectAttr`. "Driver Manager and Unicode Encoding on UNIX/Linux" describes in detail how this is done.

See also

[Driver Manager and Unicode Encoding on UNIX/Linux](#) on page 291

Unicode Application with a Non-Unicode Driver

An operation involving a Unicode application and a non-Unicode driver incurs more overhead because function conversion is involved.



Windows

1. The Unicode application sends UCS-2/UTF-16 function calls to the Driver Manager.
2. The Driver Manager converts the function calls from UCS-2/UTF-16 to ANSI. The type of ANSI is determined by the Driver Manager through reference to the client machine's Active Code Page.
3. The Driver Manager sends the ANSI function calls to the non-Unicode driver.

4. The driver returns ANSI argument values to the Driver Manager.
5. The Driver Manager converts the function calls from ANSI to UCS-2/UTF-16 and returns these converted calls to the application.

UNIX[®] UNIX and Linux

1. The Unicode application sends function calls to the Driver Manager. The Driver Manager expects the string arguments in these function calls to be UTF-8 or UTF-16 based on the value of the `SQL_ATTR_APP_UNICODE_TYPE` attribute. Note that the `SQL_ATTR_APP_UNICODE_TYPE` attribute can be set for the environment, using `SQLSetEnvAttr`, or the connection, using `SQLSetConnectAttr`.
2. The Driver Manager converts the function calls from UTF-8 or UTF-16 to ANSI. The type of ANSI is determined by the Driver Manager through reference to the client machine's value for the `IANAAppCodePage` connection string attribute.
3. The Driver Manager sends the converted ANSI function calls to the non-Unicode driver.
4. The driver returns ANSI argument values to the Driver Manager.
5. The Driver Manager converts the function calls from ANSI to UTF-8 or UTF-16 and returns these converted calls to the application.



MacOS macOS

On macOS, this scenario does not apply. All currently available drivers are Unicode drivers.

Unicode Application with a Unicode Driver

An operation involving a Unicode application and a Unicode driver that use the same Unicode encoding is efficient because no function conversion is involved. If the application and the driver each use different types of encoding, there is some conversion overhead. See "Driver Manager and Unicode Encoding on UNIX/Linux" for details.



Windows

1. The Unicode application sends UCS-2 or UTF-16 function calls to the Driver Manager.
2. The Driver Manager does not have to convert the UCS-2/UTF-16 function calls to ANSI. It passes the Unicode function call to the Unicode driver.
3. The driver returns UCS-2/UTF-16 argument values to the Driver Manager.
4. The Driver Manager returns UCS-2/UTF-16 function calls to the application.

UNIX[®] UNIX and Linux

1. The Unicode application sends function calls to the Driver Manager. The Driver Manager expects the string arguments in these function calls to be UTF-8 or UTF-16 based on the value of the `SQL_ATTR_APP_UNICODE_TYPE` attribute. Note that the `SQL_ATTR_APP_UNICODE_TYPE` attribute can be set for the environment, using `SQLSetEnvAttr`, or the connection, using `SQLSetConnectAttr`.
2. The Driver Manager passes Unicode function calls to the Unicode driver. The Driver Manager has to perform function call conversions if the `SQL_ATTR_APP_UNICODE_TYPE` is different from the `SQL_ATTR_DRIVER_UNICODE_TYPE`.

-
3. The driver returns argument values to the Driver Manager. Whether these are UTF-8 or UTF-16 argument values is based on the value of the `SQL_ATTR_DRIVER_UNICODE_TYPE` attribute.
 4. The Driver Manager returns appropriate function calls to the application based on the `SQL_ATTR_APP_UNICODE_TYPE` attribute value. The Driver Manager has to perform function call conversions if the `SQL_ATTR_DRIVER_UNICODE_TYPE` value is different from the `SQL_ATTR_APP_UNICODE_TYPE` value.



MacOS macOS

1. The Unicode application sends UTF-32 function calls to the Driver Manager.
2. The Driver Manager does not have to convert the UTF-32 function calls to ANSI. It passes the Unicode function call to the Unicode driver.
3. The driver returns UTF-32 argument values to the Driver Manager.
4. The Driver Manager returns UTF-32 function calls to the application.

See also

[Driver Manager and Unicode Encoding on UNIX/Linux](#) on page 291

Data

ODBC C data types are used to indicate the type of C buffers that store data in the application. This is in contrast to SQL data types, which are mapped to native database types to store data in a database (data store). ANSI applications bind to the C data type `SQL_C_CHAR` and expect to receive information bound in the same way. Similarly, most Unicode applications bind to the C data type `SQL_C_WCHAR` (wide data type) and expect to receive information bound in the same way. Any ODBC 3.5-compliant Unicode driver must be capable of supporting `SQL_C_CHAR` and `SQL_C_WCHAR` so that it can return data to both ANSI and Unicode applications.

When the driver communicates with the database, it must use ODBC SQL data types, such as `SQL_CHAR` and `SQL_WCHAR`, that map to native database types. In the case of ANSI data and an ANSI database, the driver receives data bound to `SQL_C_CHAR` and passes it to the database as `SQL_CHAR`. The same is true of `SQL_C_WCHAR` and `SQL_WCHAR` in the case of Unicode data and a Unicode database.

When data from the application and the data stored in the database differ in format, for example, ANSI application data and Unicode database data, conversions must be performed. The driver cannot receive `SQL_C_CHAR` data and pass it to a Unicode database that expects to receive a `SQL_WCHAR` data type. The driver or the Driver Manager must be capable of converting `SQL_C_CHAR` to `SQL_WCHAR`, and vice versa.

The simplest cases of data communication are when the application, the driver, and the database are all of the same type and encoding, ANSI-to-ANSI-to-ANSI or Unicode-to-Unicode-to-Unicode. There is no data conversion involved in these instances.

When a difference exists between data types, a conversion from one type to another must take place at the driver or Driver Manager level, which involves additional overhead. The type of driver determines whether these conversions are performed by the driver or the Driver Manager. "Driver Manager and Unicode Encoding on UNIX/Linux" describes how the Driver Manager determines the type of Unicode encoding of the application and driver.

The following sections discuss two basic types of data conversion in the Progress DataDirect *for* ODBC driver and the Driver Manager. How an individual driver exchanges different types of data with a particular database at the database level is beyond the scope of this discussion.

See also

[Driver Manager and Unicode Encoding on UNIX/Linux](#) on page 291

Unicode Driver

The Unicode driver, not the Driver Manager, must convert SQL_C_CHAR (ANSI) data to SQL_WCHAR (Unicode) data, and vice versa, as well as SQL_C_WCHAR (Unicode) data to SQL_CHAR (ANSI) data, and vice versa.

The driver must use client code page information (Active Code Page on Windows and IANAAppCodePage attribute on UNIX/Linux/macOS) to determine which ANSI code page to use for the conversions. The Active Code Page or IANAAppCodePage must match the database default character encoding; if it does not, conversion errors are possible.

ANSI Driver

The Driver Manager, not the ANSI driver, must convert SQL_C_WCHAR (Unicode) data to SQL_CHAR (ANSI) data, and vice versa (see "Unicode Support in ODBC" for a detailed discussion). This is necessary because ANSI drivers do not support any Unicode ODBC types.

The Driver Manager must use client code page information (Active Code Page on Windows and the IANAAppCodePage attribute on UNIX/Linux/macOS) to determine which ANSI code page to use for the conversions. The Active Code Page or IANAAppCodePage must match the database default character encoding. If not, conversion errors are possible.

See also

[Unicode Support in ODBC](#) on page 286

Default Unicode Mapping

The following table shows the default Unicode mapping for an application's SQL_C_WCHAR variables.

Platform	Default Unicode Mapping
Windows	UCS-2/UTF-16
AIX	UTF-8
HP-UX	UTF-8
Solaris	UTF-8
Linux	UTF-8
macOS	UTF-32

Connection Attribute for Unicode

If you do not want to use the default Unicode mappings for SQL_C_WCHAR, a connection attribute is available to override the default mappings. This attribute determines how character data is converted and presented to an application and the database.

Attribute	Description
SQL_ATTR_APP_WCHAR_TYPE (1061)	Sets the SQL_C_WCHAR type for parameter and column binding to the Unicode type, either SQL_DD_CP_UTF16 (default for Windows) or SQL_DD_CP_UTF8 (default for UNIX/Linux).

You can set this attribute before or after you connect. After this attribute is set, all conversions are made based on the character set specified.

For example:

```
rc = SQLSetConnectAttr (hdbc, SQL_ATTR_APP_WCHAR_TYPE,  
(void *)SQL_DD_CP_UTF16, SQL_IS_INTEGER);
```

SQLGetConnectAttr and SQLSetConnectAttr for the SQL_ATTR_APP_WCHAR_TYPE attribute return a SQL State of HYC00 for drivers that do not support Unicode.

This connection attribute and its valid values can be found in the file `qesqlext.h`, which is installed with the product.

Note: On Mac Platforms, the iODBC Driver Manager supports only UTF-32. As a result, this attribute is not currently supported.

Driver Manager and Unicode Encoding on UNIX/Linux

UNIX[®] Unicode ODBC drivers on UNIX and Linux can use UTF-8 or UTF-16 encoding. To use a single UTF-8 or UTF-16 application with either a UTF-8 or UTF-16 driver, the Driver Manager must be able to determine with which type of encoding the application and driver use and, if necessary, convert them accordingly.

To make this determination, the Driver Manager supports a set of ODBC attributes that can be set for the environment or the connection. If your application uses both UTF-8 and UTF-16 drivers in the same environment, encoding should be set for the connection only; otherwise, either method can be used.

- To configure the encoding type for the environment, set the ODBC environment attributes SQL_ATTR_APP_UNICODE_TYPE and SQL_ATTR_DRIVER_UNICODE_TYPE using SQLSetEnvAttr.
- To configure the encoding for the connection only, set the ODBC connection attribute SQL_ATTR_APP_UNICODE_TYPE and SQL_ATTR_DRIVER_UNICODE_TYPE using SQLSetConnectAttr.

The attributes support values of SQL_DD_CP_UTF8 and SQL_DD_CP_UTF16. The default value is SQL_DD_CP_UTF8.

Note: You must specify a value for SQL_ATTR_DRIVER_UNICODE_TYPE when using third-party drivers. However, for DataDirect drivers, the driver manager detects the Unicode type for the driver by default.

The Driver Manager performs the following steps before actually connecting to the driver.

1. Determine the application Unicode type: Applications that use UTF-16 encoding for their string types need to set SQL_ATTR_APP_UNICODE_TYPE accordingly at connection, or, if setting the encoding type for the environment, before connecting to any driver. When the Driver Manager reads this attribute, it expects all string arguments to the ODBC "W" functions to be in the specified Unicode format. This attribute also indicates how the SQL_C_WCHAR buffers must be encoded.
2. Determine the driver Unicode type: The Driver Manager must determine through which Unicode encoding the driver supports its "W" functions. This is done as follows:
 - a. SQLGetEnvAttr(SQL_ATTR_DRIVER_UNICODE_TYPE) or SQLGetConnectAttr(SQL_ATTR_DRIVER_UNICODE_TYPE) is called in the driver by the Driver Manager. The driver, if capable, returns either SQL_DD_CP_UTF16 or SQL_DD_CP_UTF8 to indicate to the Driver Manager which encoding it expects.
 - b. If the preceding call to SQLGetEnvAttr fails, the Driver Manager looks either in the Data Source section of the `odbc.ini` specified by the connection string or in the connection string itself for a connection

option named `DriverUnicodeType`. Valid values for this option are 1 (UTF-16) or 2 (UTF-8). The Driver Manager assumes that the Unicode encoding of the driver corresponds to the value specified.

- c. If neither of the preceding attempts are successful, the Driver Manager assumes that the Unicode encoding of the driver is UTF-8.
3. Determine if the driver supports `SQL_ATTR_WCHAR_TYPE`: `SQLSetConnectAttr` (`SQL_ATTR_WCHAR_TYPE, x`) is called in the driver by the Driver Manager, where `x` is either `SQL_DD_CP_UTF8` or `SQL_DD_CP_UTF16`, depending on the value of the `SQL_ATTR_APP_UNICODE_TYPE` setting. If the driver returns any error on this call to `SQLSetConnectAttr`, the Driver Manager assumes that the driver does not support this connection attribute.

If an error occurs, the Driver Manager returns a warning. The Driver Manager does not convert all bound parameter data from the application Unicode type to the driver Unicode type specified by `SQL_ATTR_DRIVER_UNICODE_TYPE`. Neither does it convert all data bound as `SQL_C_WCHAR` to the application Unicode type specified by `SQL_ATTR_APP_UNICODE_TYPE`.

Based on the information it has gathered prior to connection, the Driver Manager either does not have to convert function calls, or, before calling the driver, it converts to either UTF-8 or UTF-16 all string arguments to calls to the ODBC "W" functions.

References

The Java Tutorials, <http://docs.oracle.com/javase/tutorial/i18n/index.html>

Unicode Support in the Solaris Operating Environment, May 2000, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303-4900

Character Encoding in the `odbc.ini` and `odbcinst.ini` Files

UNIX[®] The `odbc.ini` and `odbcinst.ini` files can use ANSI or UTF-8 encoding. To ensure encoding compatibility between these files and the application, the Driver Manager converts encoding when necessary. This allows applications with different encoding to write to or read from the `odbc.ini` or `odbcinst.ini` file using the following functions:

ANSI functions:

- `SQLWritePrivateProfileString`
- `SQLGetPrivateProfileString`

Unicode (wide or "W") functions:

- `SQLWritePrivateProfileStringW`
- `SQLGetPrivateProfileStringW`

For the Driver Manager to accomplish this task, it must determine the encoding format your application and file use. How the Driver Manager makes this determination is dependent on the encoding of the function called by the application.

When a Unicode function is called, the Driver Manager assumes that the `odbc.ini` and `odbcinst.ini` files use UTF-8 encoding, while encoding for the application is determined by the `ODBC_App_Unicode_Type` variable in the system environment:

- If the variable is set to `ODBC_App_Unicode_Type=1`, the Driver Manager expects that application uses input and output strings encoded as UTF-16. When the application calls `SQLWritePrivateProfileStringW`, the Driver Manager converts UTF-16 input strings and writes them as UTF-8 in the file. When the application calls `SQLGetPrivateProfileStringW`, the Driver Manager returns the requested values using UTF-16 encoding.
- If any other value is specified for `ODBC_App_Unicode_Type`, or if the variable is not defined, the Driver Manager assumes that the application and file use UTF-8. When this occurs, the Driver Manager does not convert strings passed between the application and file.

When an ANSI function is called, the Driver Manager assumes that application and file use ANSI encoding. In this scenario, the Driver Manger does not convert strings passed between the application and file.

For more information about the `odbc.ini` and `odbcinst.ini` files, see "Configuring the Product on UNIX/Linux."

See also

[Configuring the Product on UNIX/Linux](#) on page 56

Designing ODBC Applications for Performance Optimization

Developing performance-oriented ODBC applications is not easy. Microsoft's *ODBC Programmer's Reference* does not provide information about system performance. In addition, ODBC drivers and the ODBC driver manager do not return warnings when applications run inefficiently. This chapter contains some general guidelines that have been compiled by examining the ODBC implementations of numerous shipping ODBC applications. These guidelines include:

- Use catalog functions appropriately
- Retrieve only required data
- Select functions that optimize performance
- Manage connections and updates

Following these general rules will help you solve some common ODBC performance problems, such as those listed in the following table.

Table 20: Common Performance Problems Using ODBC Applications

Problem	Solution	See guidelines in...
Network communication is slow.	Reduce network traffic.	"Using Catalog Functions"
The process of evaluating complex SQL queries on the database server is slow and can reduce concurrency.	Simplify queries.	"Using Catalog Functions" "Selecting ODBC Functions"

Problem	Solution	See guidelines in...
Excessive calls from the application to the driver slow performance.	Optimize application-to-driver interaction.	"Retrieving Data" "Selecting ODBC Functions"
Disk I/O is slow.	Limit disk input/output.	"Managing Connections and Updates"

For details, see the following topics:

- [Using Catalog Functions](#)
- [Retrieving Data](#)
- [Selecting ODBC Functions](#)
- [Managing Connections and Updates](#)

Using Catalog Functions

Because catalog functions, such as those listed here, are slow compared to other ODBC functions, their frequent use can impair system performance:

- SQLColumns
- SQLForeignKeys
- SQLGetTypeInfo
- SQLSpecialColumns
- SQLStatistics
- SQLTables

SQLGetTypeInfo is included in this list of expensive ODBC functions because many drivers must query the server to obtain accurate information about which types are supported (for example, to find dynamic types such as user defined types).

Caching Information to Minimize the Use of Catalog Functions

To return all result column information mandated by the ODBC specification, a driver may have to perform multiple queries, joins, subqueries, or unions to return the required result set for a single call to a catalog function. These particular elements of the SQL language are performance expensive.

Although it is almost impossible to write an ODBC application without catalog functions, their use should be minimized. By caching information, applications can avoid multiple executions.

For example, call SQLGetTypeInfo once in the application and cache the elements of the result set that your application depends on. It is unlikely that any application uses all elements of the result set generated by a catalog function, so the cached information should not be difficult to maintain.

Avoiding Search Patterns

Passing NULL arguments or search patterns to catalog functions generates time-consuming queries. In addition, network traffic potentially increases because of unwanted results. Always supply as many non-NULL arguments to catalog functions as possible. Because catalog functions are slow, applications should invoke them efficiently. Any information that the application can send the driver when calling catalog functions can result in improved performance and reliability.

For example, consider a call to `SQLTables` where the application requests information about the table "Customers." Often, this call is coded as shown, using as many NULL arguments as possible:

```
rc = SQLTables (hstmt, NULL, 0, NULL, 0, "Customers", SQL_NTS, NULL, 0);
```

A driver processes this `SQLTables` call into SQL that looks like this:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers'
UNION ALL
SELECT ... FROM SysViews WHERE ViewName = 'Customers'
UNION ALL
SELECT ... FROM SysSynonyms WHERE SynName = 'Customers' ORDER BY ...
```

In our example, the application provides scant information about the object for which information was requested. Suppose three "Customers" tables were returned in the result set: the first table owned by the user named Beth, the second owned by the sales department, and the third a view created by management.

It may not be obvious to the end user which table to choose. If the application had specified the `OwnerName` argument in the `SQLTables` call, only one table would be returned and performance would improve. Less network traffic would be required to return only one result row and unwanted rows would be filtered by the database. In addition, if the `TableType` argument was supplied, the SQL sent to the server can be optimized from a three-query union into a single Select statement as shown:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers' AND Owner = 'Beth'
```

Using a Dummy Query to Determine Table Characteristics

Avoid using `SQLColumns` to determine characteristics about a table. Instead, use a dummy query with `SQLDescribeCol`.

Consider an application that allows the user to choose the columns that will be selected. Should the application use `SQLColumns` to return information about the columns to the user or prepare a dummy query and call `SQLDescribeCol`?

Case 1: `SQLColumns` Method

```
rc = SQLColumns (... "UnknownTable" ...);
// This call to SQLColumns will generate a query to the system catalogs...
// possibly a join which must be prepared, executed, and produce a result set
rc = SQLBindCol (...);
rc = SQLExtendedFetch (...);
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained
```

Case 2: `SQLDescribeCol` Method

```
// prepare dummy query
rc = SQLPrepare (... "SELECT * FROM UnknownTable WHERE 1 = 0" ...);
// query is never executed on the server - only prepared
rc = SQLNumResultCols (...);
for (irow = 1; irow <= NumColumns; irow++) {
    rc = SQLDescribeCol (...);
    // + optional calls to SQLColAttributes
```

```
}  
// result column information has now been obtained  
// Note we also know the column ordering within the table!  
// This information cannot be assumed from the SQLColumns example.
```

In both cases, a query is sent to the server, but in Case 1, the query must be evaluated and form a result set that must be sent to the client. Clearly, Case 2 is the better performing model.

To complicate this discussion, let us consider a database server that does not natively support preparing a SQL statement. The performance of Case 1 does not change, but the performance of Case 2 improves slightly because the dummy query is evaluated before being prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and should execute without accessing table data. Again, for this situation, Case 2 outperforms Case 1.

Retrieving Data

To retrieve data efficiently, return only the data that you need, and choose the most efficient method of doing so. The guidelines in this section will help you optimize system performance when retrieving data with ODBC applications.

Retrieving Long Data

Because retrieving long data across the network is slow and resource-intensive, applications should not request long data (SQL_LONGVARCHAR, SQL_WLONGVARCHAR, and SQL_LONGVARBINARY data) unless it is necessary.

Most users do not want to see long data. If the user does need to see these result items, the application can query the database again, specifying only long columns in the select list. This technique allows the average user to retrieve the result set without having to pay a high performance penalty for network traffic.

Although the best approach is to exclude long data from the select list, some applications do not formulate the select list before sending the query to the ODBC driver (that is, some applications simply `SELECT * FROM table_name ...`). If the select list contains long data, the driver must retrieve that data at fetch time even if the application does not bind the long data in the result set. When possible, use a technique that does not retrieve all columns of the table.

Reducing the Size of Data Retrieved

Sometimes, long data must be retrieved. When this is the case, remember that most users do not want to see 100 KB, or more, of text on the screen.

To reduce network traffic and improve performance, you can reduce the size of data being retrieved to some manageable limit by calling `SQLSetStmtAttr` with the `SQL_ATTR_MAX_LENGTH` option.

Eliminating `SQL_LONGVARCHAR`, `SQL_WLONGVARCHAR`, and `SQL_LONGVARBINARY` data from the result set is ideal for optimizing performance.

Many application developers mistakenly assume that if they call `SQLGetData` with a container of size *x* that the ODBC driver only retrieves *x* bytes of information from the server. Because `SQLGetData` can be called multiple times for any one column, most drivers optimize their network use by retrieving long data in large chunks and then returning it to the user when requested. For example:

```
char CaseContainer[1000];  
...  
rc = SQLExecDirect (hstmt, "SELECT CaseHistory FROM Cases WHERE CaseNo = 71164", SQL_NTS);  
...  
rc = SQLFetch (hstmt);  
rc = SQLGetData (hstmt, 1, CaseContainer,(SWORD) sizeof(CaseContainer), ...);
```

At this point, it is more likely that an ODBC driver will retrieve 64 KB of information from the server instead of 1 KB. In terms of network access, one 64-KB retrieval is less expensive than 64 retrievals of 1 KB. Unfortunately, the application may not call `SQLGetData` again; therefore, the first and only retrieval of `CaseHistory` would be slowed by the fact that 64 KB of data must be sent across the network.

Many ODBC drivers allow you to limit the amount of data retrieved across the network by supporting the `SQL_MAX_LENGTH` attribute. This attribute allows the driver to communicate to the database server that only `x` bytes of data are relevant to the client. The server responds by sending only the first `x` bytes of data for all result columns. This optimization substantially reduces network traffic and improves client performance. The previous example returned only one row, but consider the case where 100 rows are returned in the result set—the performance improvement would be substantial.

Using Bound Columns

Retrieving data through bound columns (`SQLBindCol`) instead of using `SQLGetData` reduces the ODBC call load and improves performance.

Consider the following code fragment:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM Employees WHERE HireDate >= ?", SQL_NTS);
do {
    rc = SQLFetch (hstmt);
    // call SQLGetData 20 times
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

Suppose the query returns 90 result rows. In this case, 1891 ODBC calls are made (20 calls to `SQLGetData` x 90 result rows + 91 calls to `SQLFetch`).

Consider the same scenario that uses `SQLBindCol` instead of `SQLGetData`:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM Employees WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 20 times
do {
    rc = SQLFetch (hstmt);
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made is reduced from 1891 to 111 (20 calls to `SQLBindCol` + 91 calls to `SQLFetch`). In addition to reducing the call load, many drivers optimize how `SQLBindCol` is used by binding result information directly from the database server into the user's buffer. That is, instead of the driver retrieving information into a container and then copying that information to the user's buffer, the driver simply requests the information from the server be placed directly into the user's buffer.

Using `SQLExtendedFetch` Instead of `SQLFetch`

Use `SQLExtendedFetch` to retrieve data instead of `SQLFetch`. The ODBC call load decreases (resulting in better performance) and the code is less complex (resulting in more maintainable code).

Most ODBC drivers now support `SQLExtendedFetch` for forward only cursors; yet, most ODBC applications use `SQLFetch` to retrieve data. Consider the examples in "Using Bound Columns", this time using `SQLExtendedFetch` instead of `SQLFetch`:

```
rc = SQLSetStmtOption (hstmt, SQL_ROWSET_SIZE, 100);
// use arrays of 100 elements
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM Employees WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 1 time specifying row-wise binding
do {
    rc = SQLExtendedFetch (hstmt, SQL_FETCH_NEXT, 0, &RowsFetched, RowStatus);
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

Notice the improvement from the previous examples. The initial call load was 1891 ODBC calls. By choosing ODBC calls carefully, the number of ODBC calls made by the application has now been reduced to 4 (1 `SQLSetStmtOption` + 1 `SQLExecDirect` + 1 `SQLBindCol` + 1 `SQLExtendedFetch`). In addition to reducing the call load, many ODBC drivers retrieve data from the server in arrays, further improving the performance by reducing network traffic.

For ODBC drivers that do not support `SQLExtendedFetch`, the application can enable forward-only cursors using the ODBC cursor library:

```
(rc=SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS, SQL_CUR_USE_IF_NEEDED);
```

Although using the cursor library does not improve performance, it should not be detrimental to application response time when using forward-only cursors (no logging is required). Furthermore, using the cursor library means that the application can always depend on `SQLExtendedFetch` being available. This simplifies the code because the application does not require two algorithms (one using `SQLExtendedFetch` and one using `SQLFetch`).

See also

[Using Bound Columns](#) on page 299

Choosing the Right Data Type

Retrieving and sending certain data types can be expensive. When you are working with data on a large scale, select the data type that can be processed most efficiently. For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

Selecting ODBC Functions

The guidelines in this section will help you select which ODBC functions will give you the best performance.

Using SQLPrepare/SQLExecute and SQLExecDirect

Using `SQLPrepare/SQLExecute` is not always as efficient as `SQLExecDirect`. Use `SQLExecDirect` for queries that will be executed once and `SQLPrepare/SQLExecute` for queries that will be executed multiple times.

ODBC drivers are optimized based on the perceived use of the functions that are being executed. `SQLPrepare/SQLExecute` is optimized for multiple executions of statements that use parameter markers. `SQLExecDirect` is optimized for a single execution of a SQL statement. Unfortunately, more than 75% of all ODBC applications use `SQLPrepare/SQLExecute` exclusively.

Consider the case where an ODBC driver implements `SQLPrepare` by creating a stored procedure on the server that contains the prepared statement. Creating stored procedures involve substantial overhead, but the statement can be executed multiple times. Although creating stored procedures is performance-expensive, execution is minimal because the query is parsed and optimization paths are stored at create procedure time.

Using `SQLPrepare/SQLExecute` for a statement that is executed only once results in unnecessary overhead. Furthermore, applications that use `SQLPrepare/SQLExecute` for large single execution query batches exhibit poor performance. Similarly, applications that always use `SQLExecDirect` do not perform as well as those that use a logical combination of `SQLPrepare/SQLExecute` and `SQLExecDirect` sequences.

Using Arrays of Parameters

Passing arrays of parameter values for bulk insert operations, for example, with `SQLPrepare/SQLExecute` and `SQLExecDirect` can reduce the ODBC call load and network traffic. To use arrays of parameters, the application calls `SQLSetStmtAttr` with the following attribute arguments:

- `SQL_ATTR_PARAMSET_SIZE` sets the array size of the parameter.

- SQL_ATTR_PARAMS_PROCESSED_PTR assigns a variable filled by SQLExecute, which contains the number of rows that are actually inserted.
- SQL_ATTR_PARAM_STATUS_PTR points to an array in which status information for each row of parameter values is returned.

Note: ODBC 3.x replaced the ODBC 2.x call to SQLParamOptions with calls to SQLSetStmtAttr using the SQL_ATTR_PARAMSET_SIZE, SQL_ATTR_PARAMS_PROCESSED_ARRAY, and SQL_ATTR_PARAM_STATUS_PTR arguments.

Before executing the statement, the application sets the value of each data element in the bound array. When the statement is executed, the driver tries to process the entire array contents using one network roundtrip. For example, let us compare the following examples, Case 1 and Case 2.

Case 1: Executing Prepared Statement Multiple Times

```
rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger (...) VALUES (?, ?, ...)", SQL_NTS);
// bind parameters
...
do {
    // read ledger values into bound parameter buffers
    ...
    rc = SQLExecute (hstmt);
    // insert row
} while ! (eof);
```

Case 2: Using Arrays of Parameters

```
SQLPrepare (hstmt, " INSERT INTO DailyLedger (...) VALUES (?, ?, ...)", SQL_NTS);
SQLSetStmtAttr (hstmt, SQL_ATTR_PARAMSET_SIZE, (UDWORD)100, SQL_IS_UIINTEGER);
SQLSetStmtAttr (hstmt, SQL_ATTR_PARAMS_PROCESSED_PTR, &rows_processed, SQL_IS_POINTER);
// Specify an array in which to return the status of
// each set of parameters.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAM_STATUS_PTR, ParamStatusArray, SQL_IS_POINTER);
// pass 100 parameters per execute
// bind parameters
...
do {
    // read up to 100 ledger values into
    // bound parameter buffers
    ...
    rc = SQLExecute (hstmt);
    // insert a group of 100 rows
} while ! (eof);
```

In Case 1, if there are 100 rows to insert, 101 network roundtrips are required to the server, one to prepare the statement with SQLPrepare and 100 additional roundtrips for each time SQLExecute is called.

In Case 2, the call load has been reduced from 100 SQLExecute calls to only 1 SQLExecute call. Furthermore, network traffic is reduced considerably.

Using the Cursor Library

If the driver provides scrollable cursors, do not use the cursor library. The cursor library creates local temporary log files, which are performance-expensive to generate and provide worse performance than native scrollable cursors.

The cursor library adds support for static cursors, which simplifies the coding of applications that use scrollable cursors. However, the cursor library creates temporary log files on the user's local disk drive to accomplish the task. Typically, disk I/O is a slow operation. Although the cursor library is beneficial, applications should not automatically choose to use the cursor library when an ODBC driver supports scrollable cursors natively.

Typically, ODBC drivers that support scrollable cursors achieve high performance by requesting that the database server produce a scrollable result set instead of emulating the capability by creating log files. Many applications use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS, SQL_CUR_USE_ODBC);
```

but should use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS, SQL_CUR_USE_IF_NEEDED);
```

Managing Connections and Updates

The guidelines in this section will help you to manage connections and updates to improve system performance for your ODBC applications.

Managing Connections

Connection management is important to application performance. Optimize your application by connecting once and using multiple statement handles, instead of performing multiple connections. Avoid connecting to a data source after establishing an initial connection.

Although gathering driver information at connect time is a good practice, it is often more efficient to gather it in one step rather than two steps. Some ODBC applications are designed to call informational gathering routines that have no record of already attached connection handles. For example, some applications establish a connection and then call a routine in a separate DLL or shared library that reattaches and gathers information about the driver. Applications that are designed as separate entities should pass the already connected HDBC pointer to the data collection routine instead of establishing a second connection.

Another bad practice is to connect and disconnect several times throughout your application to process SQL statements. Connection handles can have multiple statement handles associated with them. Statement handles can provide memory storage for information about SQL statements. Therefore, applications do not need to allocate new connection handles to process SQL statements. Instead, applications should use statement handles to manage multiple SQL statements.

You can significantly improve performance with connection pooling, especially for applications that connect over a network or through the World Wide Web. With connection pooling, closing connections does not close the physical connection to the database. When an application requests a connection, an active connection from the connection pool is reused, avoiding the network round trips needed to create a new connection.

Connection and statement handling should be addressed before implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

Managing Commits in Transactions

Committing data is extremely disk I/O intensive and slow. If the driver can support transactions, always turn autocommit off.

What does a commit actually involve? The database server must flush back to disk every data page that contains updated or new data. This is not a sequential write but a searched write to replace existing data in the table. By default, autocommit is on when connecting to a data source. Autocommit mode usually impairs system performance because of the significant amount of disk I/O needed to commit every operation.

Some database servers do not provide an Autocommit mode. For this type of server, the ODBC driver must explicitly issue a COMMIT statement and a BEGIN TRANSACTION for every operation sent to the server. In addition to the large amount of disk I/O required to support Autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

Although using transactions can help application performance, do not take this tip too far. Leaving transactions active can reduce throughput by holding locks on rows for long times, preventing other users from accessing the rows. Commit transactions in intervals that allow maximum concurrency.

Choosing the Right Transaction Model

Many systems support distributed transactions; that is, transactions that span multiple connections. Distributed transactions are at least four times slower than normal transactions due to the logging and network round trips necessary to communicate between all the components involved in the distributed transaction. Unless distributed transactions are required, avoid using them. Instead, use local transactions when possible.

Using Positioned Updates and Deletes

Use positioned updates and deletes or `SQLSetPos` to update data. Although positioned updates do not apply to all types of applications, developers should use positioned updates and deletes when it makes sense. Positioned updates (either through `UPDATE WHERE CURRENT OF CURSOR` or through `SQLSetPos`) allow the developer to signal the driver to "change the data here" by positioning the database cursor at the appropriate row to be changed. The designer is not forced to build a complex SQL statement, but simply supplies the data to be changed.

In addition to making the application more maintainable, positioned updates usually result in improved performance. Because the database server is already positioned on the row for the `Select` statement in process, performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server typically has an internal pointer to the row available (for example, `ROWID`).

Using `SQLSpecialColumns`

Use `SQLSpecialColumns` to determine the optimal set of columns to use in the `Where` clause for updating data. Often, pseudo-columns provide the fastest access to the data, and these columns can only be determined by using `SQLSpecialColumns`.

Some applications cannot be designed to take advantage of positioned updates and deletes. These applications typically update data by forming a `Where` clause consisting of some subset of the column values returned in the result set. Some applications may formulate the `Where` clause by using all searchable result columns or by calling `SQLStatistics` to find columns that are part of a unique index. These methods typically work, but can result in fairly complex queries.

Consider the following example:

```
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name, ssn, address, city, state, zip
  FROM emp", SQL_NTS);
// fetchdata
...
rc = SQLExecDirect (hstmt, "UPDATE EMP SET ADDRESS = ? WHERE first_name = ? AND last_name
  = ? AND
  ssn = ? AND address = ? AND city = ? AND state = ? AND zip = ?", SQL_NTS);
// fairly complex query
```

Applications should call `SQLSpecialColumns/SQL_BEST_ROWID` to retrieve the optimal set of columns (possibly a pseudo-column) that identifies a given record. Many databases support special columns that are not explicitly defined by the user in the table definition but are "hidden" columns of every table (for example, `ROWID` and `TID`). These pseudo-columns provide the fastest access to data because they typically point to the exact location of the record. Because pseudo-columns are not part of the explicit table definition, they are not returned from `SQLColumns`. To determine if pseudo-columns exist, call `SQLSpecialColumns`.

Consider the previous example again:

```
...
rc = SQLSpecialColumns (hstmt, ..... 'emp', ...);
...
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name, ssn, address, city, state,
zip, ROWID
    FROM emp", SQL_NTS);
// fetch data and probably "hide" ROWID from the user
...
rc = SQLExecDirect (hstmt, "UPDATE emp SET address = ? WHERE ROWID = ?",SQL_NTS);
// fastest access to the data!
```

If your data source does not contain special pseudo-columns, the result set of `SQLSpecialColumns` consists of columns of the optimal unique index on the specified table (if a unique index exists). Therefore, your application does not need to call `SQLStatistics` to find the smallest unique index.

Locking and Isolation Levels

This chapter discusses locking and isolation levels and how their settings can affect the data you retrieve. Oracle supports isolation level 1 (read committed) and isolation level 3 (serializable). Oracle supports record-level locking.

For details, see the following topics:

- [Locking](#)
- [Isolation Levels](#)
- [Locking Modes and Levels](#)

Locking

Locking is a database operation that restricts a user from accessing a table or record. Locking is used in situations where more than one user might try to use the same table or record at the same time. By locking the table or record, the system ensures that only one user at a time can affect the data.

Locking is critical in multiuser databases, where different users can try to access or modify the same records concurrently. Although such concurrent database activity is desirable, it can create problems. Without locking, for example, if two users try to modify the same record at the same time, they might encounter problems ranging from retrieving bad data to deleting data that the other user needs. If, however, the first user to access a record can lock that record to temporarily prevent other users from modifying it, such problems can be avoided. Locking provides a way to manage concurrent database access while minimizing the various problems it can cause.

Isolation Levels

An isolation level represents a particular locking strategy employed in the database system to improve data consistency. The higher the isolation level, the more complex the locking strategy behind it. The isolation level provided by the database determines whether a transaction will encounter the following behaviors in data consistency:

Dirty reads	User 1 modifies a row. User 2 reads the same row before User 1 commits. User 1 performs a rollback. User 2 has read a row that has never really existed in the database. User 2 may base decisions on false data.
Non-repeatable reads	User 1 reads a row, but does not commit. User 2 modifies or deletes the same row and then commits. User 1 rereads the row and finds it has changed (or has been deleted).
Phantom reads	User 1 uses a search condition to read a set of rows, but does not commit. User 2 inserts one or more rows that satisfy this search condition, then commits. User 1 rereads the rows using the search condition and discovers rows that were not present before.

Isolation levels represent the database system's ability to prevent these behaviors. The American National Standards Institute (ANSI) defines four isolation levels:

- Read uncommitted (0)
- Read committed (1)
- Repeatable read (2)
- Serializable (3)

In ascending order (0–3), these isolation levels provide an increasing amount of data consistency to the transaction. At the lowest level, all three behaviors can occur. At the highest level, none can occur. The success of each level in preventing these behaviors is due to the locking strategies they use, which are as follows:

Read uncommitted (0)	Locks are obtained on modifications to the database and held until end of transaction (EOT). Reading from the database does not involve any locking.
Read committed (1)	Locks are acquired for reading and modifying the database. Locks are released after reading but locks on modified objects are held until EOT.
Repeatable read (2)	Locks are obtained for reading and modifying the database. Locks on all modified objects are held until EOT. Locks obtained for reading data are held until EOT. Locks on non-modified access structures (such as indexes and hashing structures) are released after reading.
Serializable (3)	All data read or modified is locked until EOT. All access structures that are modified are locked until EOT. Access structures used by the query are locked until EOT.

The following table shows what data consistency behaviors can occur at each isolation level.

Table 21: Isolation Levels and Data Consistency

Level	Dirty Read	Nonrepeatable Read	Phantom Read
0, Read uncommitted	Yes	Yes	Yes
1, Read committed	No	Yes	Yes

Level	Dirty Read	Nonrepeatable Read	Phantom Read
2, Repeatable read	No	No	Yes
3, Serializable	No	No	No

Although higher isolation levels provide better data consistency, this consistency can be costly in terms of the concurrency provided to individual users. Concurrency is the ability of multiple users to access and modify data simultaneously. As isolation levels increase, so does the chance that the locking strategy used will create problems in concurrency.

The higher the isolation level, the more locking involved, and the more time users may spend waiting for data to be freed by another user. Because of this inverse relationship between isolation levels and concurrency, you must consider how people use the database before choosing an isolation level. You must weigh the trade-offs between data consistency and concurrency, and decide which is more important.

Locking Modes and Levels

Different database systems use various locking modes, but they have two basic modes in common: shared and exclusive. Shared locks can be held on a single object by multiple users. If one user has a shared lock on a record, then a second user can also get a shared lock on that same record; however, the second user cannot get an exclusive lock on that record. Exclusive locks are exclusive to the user that obtains them. If one user has an exclusive lock on a record, then a second user cannot get either type of lock on the same record.

Performance and concurrency can also be affected by the locking level used in the database system. The locking level determines the size of an object that is locked in a database. For example, many database systems let you lock an entire table, as well as individual records. An intermediate level of locking, page-level locking, is also common. A page contains one or more records and is typically the amount of data read from the disk in a single disk access. The major disadvantage of page-level locking is that if one user locks a record, a second user may not be able to lock other records because they are stored on the same page as the locked record.

SSL Encryption Cipher Suites

See "Using Security" for information about using Secure Sockets Layer (SSL) data encryption with the drivers. Transport Layer Security (TLS) protocols are supported as listed in this chapter.

The following tables list the SSL and encryption cipher suites supported by your Progress DataDirect *for* ODBC driver. The driver attempts to negotiate either SSL v3 or TLS v1 with the server using OpenSSL cipher suites.

The following table shows the OpenSSL encryption cipher suites that the driver can use if it can negotiate SSL v2 with the server, with the name of the corresponding SSL v2 encryption cipher suites.

Note: OpenSSL libraries that are 1.1.1 and higher do not support SSL v2 cipher suites. For more information on specifying the OpenSSL library versions used by the driver, see "Designating an OpenSSL Library."

Table 22: OpenSSL Cipher Suites to SSL v2 Cipher Suites

OpenSSL Cipher Suite	SSL Encryption Cipher Suite
DES-CBC-MD5	SSL_CK_DES_64_CBC_WITH_MD5
DES-CBC3-MD5	SSL_CK_DES_192_EDE3_CBC_WITH_MD5
EXP-RC2-CBC-MD5	SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5
EXP-RC4-MD5	SSL_CK_RC4_128_EXPORT40_WITH_MD5
RC2-CBC-MD5	SSL_CK_RC2_128_CBC_WITH_MD5
RC4-MD5	SSL_CK_RC4_128_WITH_MD5

The following table shows the OpenSSL encryption cipher suites that the driver can use if it can negotiate SSL v3 with the server, with the name of the corresponding SSL v3 encryption cipher suites.

Table 23: Mapping OpenSSL Cipher Suites to SSL v3 Cipher Suites

OpenSSL Cipher Suite	SSL v3 Cipher Suite
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA ¹⁰
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA ¹⁰
AES256-SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256
DES-CBC3-SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
DES-CBC-SHA	SSL_RSA_WITH_DES_CBC_SHA
DHE-DSS-AES128-GCM-SHA256	TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
DHE-DSS-AES128-SHA	TLS_DHE_DSS_WITH_AES_128_CBC_SHA ¹⁰
DHE-DSS-AES128-SHA256	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
DHE-DSS-AES256-GCM-SHA384	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
DHE-DSS-AES256-SHA	TLS_DHE_DSS_WITH_AES_256_CBC_SHA ¹⁰
DHE-DSS-AES256-SHA256	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
DHE-DSS-SEED-SHA	TLS_DHE_DSS_WITH_SEED_CBC_SHA ¹¹
DHE-RSA-AES128-GCM-SHA256	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA ¹⁰
DHE-RSA-AES128-SHA256	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
DHE-RSA-AES256-GCM-SHA384	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA ¹⁰
DHE-RSA-AES256-SHA256	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
DHE-RSA-SEED-SHA	TLS_DHE_RSA_WITH_SEED_CBC_SHA ¹¹
EDH-DSS-DES-CBC3-SHA	SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

¹⁰ AES cipher suites from RFC3268 are used to extend TLS v1.

OpenSSL Cipher Suite	SSL v3 Cipher Suite
EDH-DSS-DES-CBC-SHA	SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
EDH-RSA-DES-CBC3-SHA	SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
EDH-RSA-DES-CBC-SHA	SSL_DHE_RSA_WITH_DES_CBC_SHA
EXP-DES-CBC-SHA	SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
EXP-EDH-DSS-DES-CBC-SHA	SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
EXP-EDH-RSA-DES-CBC-SHA	SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
EXP-RC2-CBC-MD5	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
EXP-RC4-MD5	SSL_RSA_EXPORT_WITH_RC4_40_MD5
PSK-3DES-EDE-CBC-SHA	TLS_PSK_WITH_3DES_EDE_CBC_SHA
PSK-AES128-CBC-SHA	TLS_PSK_WITH_AES_128_CBC_SHA
PSK-AES256-CBC-SHA	TLS_PSK_WITH_AES_256_CBC_SHA
PSK-RC4-SHA	TLS_PSK_WITH_RC4_128_SHA
RC4-MD5	SSL_RSA_WITH_RC4_128_MD5
RC4-SHA	SSL_RSA_WITH_RC4_128_SHA
SEED-SHA	TLS_RSA_WITH_SEED_CBC_SHA ¹¹
SRP-3DES-EDE-CBC-SHA	TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA
SRP-AES-128-CBC-SHA	TLS_SRP_SHA_WITH_AES_128_CBC_SHA
SRP-AES-256-CBC-SHA	TLS_SRP_SHA_WITH_AES_256_CBC_SHA
SRP-DSS-3DES-EDE-CBC-SHA	TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA
SRP-DSS-AES-128-CBC-SHA	TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA
SRP-DSS-AES-256-CBC-SHA	TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA
SRP-RSA-3DES-EDE-CBC-SHA	TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA
SRP-RSA-AES-128-CBC-SHA	TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA
SRP-RSA-AES-256-CBC-SHA	TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA

¹¹ Seed cipher suites from RFC4162 are used to extend TLS v1.

The following table shows the OpenSSL Encryption Cipher suites that the driver can use if it can negotiate TLS v1.0, TLS v1.1, and TLS v1.2 with the server, with the name of the corresponding cipher suites.

Table 24: Mapping OpenSSL Encryption Cipher Suites to TLS v1.0, TLS v1.1, and TLS v1.2 Cipher Suites

OpenSSL Cipher Suite	Maps to TLS v1 Cipher Suite
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA ¹⁰
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA ¹⁰
AES256-SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256
ARIA128-GCM-SHA256	TLS_RSA_WITH_ARIA_128_GCM_SHA256 ¹²
ARIA256-GCM-SHA384	TLS_RSA_WITH_ARIA_256_GCM_SHA384 ¹²
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
DES-CBC-SHA	TLS_RSA_WITH_DES_CBC_SHA
DHE-DSS-AES128-GCM-SHA256	DHE-DSS-AES128-GCM-SHA256
DHE-DSS-AES128-SHA	TLS_DHE_DSS_WITH_AES_128_CBC_SHA ¹⁰
DHE-DSS-AES128-SHA256	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
DHE-DSS-AES256-GCM-SHA384	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
DHE-DSS-AES256-SHA	TLS_DHE_DSS_WITH_AES_256_CBC_SHA ¹⁰
DHE-DSS-AES256-SHA256	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
DHE-DSS-ARIA128-GCM-SHA256	TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256 ¹²
DHE-DSS-ARIA256-GCM-SHA384	TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384 ¹²
DHE-DSS-SEED-SHA	TLS_DHE_DSS_WITH_SEED_CBC_SHA ¹¹
DHE-PSK-ARIA128-GCM-SHA256	TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256 ¹²
DHE-PSK-ARIA256-GCM-SHA384	TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384 ¹²
DHE-PSK-CHACHA20-POLY1305	TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256 ¹²
DHE-RSA-AES128-GCM-SHA256	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA ¹⁰

OpenSSL Cipher Suite	Maps to TLS v1 Cipher Suite
DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
DHE-RSA-AES256-GCM-SHA384	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA ¹⁰
DHE-RSA-AES256-SHA256	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
DHE-RSA-ARIA128-GCM-SHA256	TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 ¹²
DHE-RSA-ARIA256-GCM-SHA384	TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384 ¹²
DHE-RSA-CHACHA20-POLY1305	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 ¹²
DHE-RSA-SEED-SHA	TLS_DHE_RSA_WITH_SEED_CBC_SHA ¹¹
ECDHE-ARIA128-GCM-SHA256	TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 ¹²
ECDHE-ARIA256-GCM-SHA384	TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384 ¹²
ECDHE-RSA-CHACHA20-POLY1305	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 ¹²
ECDHE-ECDSA-ARIA128-GCM-SHA256	TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 ¹²
ECDHE-ECDSA-ARIA256-GCM-SHA384	TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384 ¹²
ECDHE-ECDSA-CHACHA20-POLY1305	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 ¹²
ECDHE-PSK-CHACHA20-POLY1305	TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256 ¹²
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
EDH-DSS-DES-CBC3-SHA	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
EDH-DSS-DES-CBC-SHA	TLS_DHE_DSS_WITH_DES_CBC_SHA
EDH-RSA-DES-CBC3-SHA	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
EDH-RSA-DES-CBC-SHA	TLS_DHE_RSA_WITH_DES_CBC_SHA
EXP-DES-CBC-SHA	TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
EXP-EDH-DSS-DES-CBC-SHA	TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

¹² Supported by OpenSSL libraries that are version 1.1.1 and higher.

OpenSSL Cipher Suite	Maps to TLS v1 Cipher Suite
EXP-EDH-RSA-DES-CBC-SHA	TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
EXP-RC2-CBC-MD5	TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
EXP-RC4-MD5	TLS_RSA_EXPORT_WITH_RC4_40_MD5
PSK-3DES-EDE-CBC-SHA	TLS_PSK_WITH_3DES_EDE_CBC_SHA
PSK-AES128-CBC-SHA	TLS_PSK_WITH_AES_128_CBC_SHA
PSK-AES256-CBC-SHA	TLS_PSK_WITH_AES_256_CBC_SHA
PSK-ARIA128-GCM-SHA256	TLS_PSK_WITH_ARIA_128_GCM_SHA256 ¹²
PSK-ARIA256-GCM-SHA384	TLS_PSK_WITH_ARIA_256_GCM_SHA384 ¹²
PSK-CHACHA20-POLY1305	TLS_PSK_WITH_CHACHA20_POLY1305_SHA256 ¹²
PSK-RC4-SHA	TLS_PSK_WITH_RC4_128_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5
RC4-SHA	TLS_RSA_WITH_RC4_128_SHA
RSA-PSK-ARIA128-GCM-SHA256	TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256 ¹²
RSA-PSK-ARIA256-GCM-SHA384	TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384 ¹²
RSA-PSK-CHACHA20-POLY1305	TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256 ¹²
SEED-SHA	TLS_RSA_WITH_SEED_CBC_SHA ¹¹
SRP-3DES-EDE-CBC-SHA	TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA
SRP-AES-128-CBC-SHA	TLS_SRP_SHA_WITH_AES_128_CBC_SHA
SRP-AES-128-CBC-SHA	TLS_SRP_SHA_WITH_AES_128_CBC_SHA
SRP-AES-256-CBC-SHA	TLS_SRP_SHA_WITH_AES_256_CBC_SHA
SRP-DSS-3DES-EDE-CBC-SHA	TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA
SRP-DSS-AES-128-CBC-SHA	TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA
SRP-DSS-AES-256-CBC-SHA	TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA
SRP-RSA-3DES-EDE-CBC-SHA	TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA
SRP-RSA-AES-128-CBC-SHA	TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA
SRP-RSA-AES-256-CBC-SHA	TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA

Reference:

[OpenSSL Cryptography and SSL/TLS Toolkit](#)

DataDirect Bulk Load



This section contains detailed information about the functions and statement attributes associated with DataDirect Bulk Load. The driver currently supports DataDirect Bulk Load on Windows, UNIX, and Linux platforms.

For a full discussion of the features and operation of DataDirect Bulk Load, see "Using DataDirect Bulk Load."

For details, see the following topics:

- [DataDirect Bulk Load Functions](#)
- [Utility Functions](#)
- [Export, Validate, and Load Functions](#)
- [DataDirect Bulk Load Statement Attributes](#)

DataDirect Bulk Load Functions

The following DataDirect functions and parameters are not part of the standard ODBC API. They include functions for returning errors and warnings on bulk operations as well as functions for bulk export, loading, and verification:

- `GetBulkDiagRec` and `GetBulkDiagRecW`
- `ExportTableToFile` and `ExportTableToFileW`
- `ValidateTableFromFile` and `ValidateTableFromFileW`
- `LoadTableFromFile` and `LoadTableFromFileW`

Note: For your application to use DataDirect Bulk Load functionality, it 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` source file shipped with the drivers contains the function `resolveName` that illustrates how to obtain function pointers to the bulk load functions.

All of this is detailed in the code examples shown in the following sections. All of these functions can be found in the commented `bulk.c` source file that ships with the drivers. This file 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 about the `bulk.c` file.

See also

[GetBulkDiagRec and GetBulkDiagRecW](#) on page 318

[ExportTableToFile and ExportTableToFileW](#) on page 320

[ValidateTableFromFile and ValidateTableFromFileW](#) on page 323

[LoadTableFromFile and LoadTableFromFileW](#) on page 325

Utility Functions

The example code in this section shows utility functions to which the DataDirect functions for bulk exporting, verification, and bulk loading refer, as well as the DataDirect functions `GetBulkDiagRec` and `GetBulkDiagRecW`.

GetBulkDiagRec and GetBulkDiagRecW

Syntax

```
SQLReturn
GetBulkDiagRec    (SQLSMALLINT  HandleType,
                  SQLHANDLE     Handle,
                  SQLSMALLINT  RecNumber,
                  SQLCHAR*     Sqlstate,
                  SQLINTEGER*   NativeError,
                  SQLCHAR*     MessageText,
                  SQLSMALLINT  BufferLength,
                  SQLSMALLINT* TextLength);

GetBulkDiagRecW  (SQLSMALLINT  HandleType,
                  SQLHANDLE     Handle,
                  SQLSMALLINT  RecNumber,
                  SQLWCHAR*    Sqlstate,
                  SQLINTEGER*   NativeError,
                  SQLWCHAR*    MessageText,
                  SQLSMALLINT  BufferLength,
                  SQLSMALLINT* TextLength);
```

The standard ODBC return codes are returned: `SQL_SUCCESS`, `SQL_SUCCESS_WITH_INFO`, `SQL_INVALID_HANDLE`, `SQL_NO_DATA`, and `SQL_ERROR`.

Description

GetBulkDiagRec (ANSI application) and GetBulkDiagRecW (Unicode application) return errors and warnings generated by bulk operations. The argument definition, return values, and function behavior is the same as for the standard ODBC SQLGetDiagRec and SQLGetDiagRecW functions with the following exceptions:

- The GetBulkDiagRec and GetBulkDiagRecW functions can be called after a bulk load, export or validate function is invoked to retrieve any error messages generated by the bulk operation. Calling these functions after any function except a bulk function is not recommended.
- The values returned in the Sqlstate and MessageText buffers by the GetBulkDiagRecW function are encoded as UTF-16 on Windows platforms. On UNIX and Linux platforms, the values returned for Sqlstate and MessageText are UTF-16 if the value of the SQL_ATTR_APP_UNICODE_TYPE is SQL_DD_CP_UTF16 and UTF-8 if the value of SQL_ATTR_APP_UNICODE_TYPE is SQL_DD_CP_UTF8.
- The handle passed as the Handle argument must be a driver connection handle obtained by calling SQLGetInfo (<ODBC Conn Handle>, SQL_DRIVER_HDBC).
- SQL_HANDLE_DBC is the only value accepted for HandleType. Any other value causes an error to be returned.

Example

```
#include "qesqlext.h"

#ifdef NULL
#define NULL 0
#endif

#if (! defined (_WIN32)) && (! defined (_WIN64))
typedef void * HMODULE;
#endif

/* Get the address of a routine in a shared library or DLL. */
void * resolveName (
    HMODULE hmod,
    const char *name)
{
    #if defined (_WIN32) || defined (_WIN64)
        return GetProcAddress (hmod, name);
    #else
        return dlsym (hmod, name);
    #endif
}

/* Get errors directly from the driver's connection handle. */
void driverError (void *driverHandle, HMODULE hmod)
{
    {
        UCHAR sqlstate[16];
        UCHAR errmsg[SQL_MAX_MESSAGE_LENGTH * 2];
        SDWORD nativeerr;
        SWORD actualmsglen;
        RETCODE rc;
        SQLSMALLINT i;
        PGetBulkDiagRec getBulkDiagRec;

        getBulkDiagRec = (PGetBulkDiagRec)
            resolveName (hmod, "GetBulkDiagRec");

        if (! getBulkDiagRec) {
            printf ("Cannot find GetBulkDiagRec!\n");
            return;
        }

        i = 1;
    loop:    rc = (*getBulkDiagRec) (SQL_HANDLE_DBC,
```

```

        driverHandle, i++,
        sqlstate, &nativeerr, errmsg,
        SQL_MAX_MESSAGE_LENGTH - 1, &actualmsglen);

    if (rc == SQL_ERROR) {
        printf ("GetBulkDiagRec failed!\n");
        return;
    }

    if (rc == SQL_NO_DATA_FOUND) return;

    printf ("SQLSTATE = %s\n", sqlstate);
    printf ("NATIVE ERROR = %d\n", nativeerr);
    errmsg[actualmsglen] = '\0';
    printf ("MSG = %s\n\n", errmsg);
    goto loop;
}

```

Export, Validate, and Load Functions

The example code in this section shows the DataDirect functions for bulk exporting, verification, and bulk loading.

ExportTableToFile and ExportTableToFileW

Syntax

```

SQLReturn
ExportTableToFile (HDBC      hdbc,
                  SQLCHAR*  TableName,
                  SQLCHAR*  FileName,
                  SQLLEN    IANAAppCodePage,
                  SQLLEN    ErrorTolerance,
                  SQLLEN    WarningTolerance,
                  SQLCHAR*  LogFile)
ExportTableToFileW (HDBC      hdbc,
                  SQLWCHAR*  TableName,
                  SQLWCHAR*  FileName,
                  SQLLEN    IANAAppCodePage,
                  SQLLEN    ErrorTolerance,
                  SQLLEN    WarningTolerance,
                  SQLWCHAR*  LogFile)

```

The standard ODBC return codes are returned: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE, and SQL_ERROR.

Purpose

ExportTableToFile (ANSI application) and ExportTableToFileW (Unicode application) bulk export a table to a physical file. Both a bulk data file and a bulk configuration file are produced by this operation. The configuration file has the same name as the data file, but with an XML extension. 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.

Parameters

hdbc

is the driver's connection handle, which is not the handle returned by `SQLAllocHandle` or `SQLAllocConnect`. To obtain the driver's connection handle, the application must then use the standard ODBC function `SQLGetInfo` (ODBC Conn Handle, `SQL_DRIVER_HDBC`).

TableName

is a null-terminated string that specifies the name of the source database table that contains the data to be exported.

FileName

is a null-terminated string that specifies the path (relative or absolute) and file name 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. This file must not already exist. If the file already exists, an error is returned.

IANAAppCodePage

specifies the code page value to which the driver must convert all data for storage in the bulk data file. See "Code Page Values" for details about `IANAAppCodePage`. See "Character Set Conversions" for more information.

The default value on Windows is the current code page of the machine. On UNIX, Linux, and macOS the default value is 4.

ErrorTolerance

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. `WarningTolerance` 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.

LogFile

is a null-terminated character string that specifies the path (relative or absolute) and file name of the bulk 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 `LogFile` is NULL, no log file is created.

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);
}

```

See also

[External Overflow Files](#) on page 156

[Code Page Values](#) on page 267

[Character Set Conversions](#) on page 156

ValidateTableFromFile and ValidateTableFromFileW

Syntax

```
SQLReturn
ValidateTableFromFile (HDBC      hdbc,
                      SQLCHAR*  TableName,
                      SQLCHAR*  ConfigFile,
                      SQLCHAR*  MessageList,
                      SQLULEN    MessageListSize,
                      SQLULEN*   NumMessages)

ValidateTableFromFileW (HDBC      hdbc,
                       SQLCHAR*  TableName,
                       SQLCHAR*  ConfigFile,
                       SQLCHAR*  MessageList,
                       SQLULEN    MessageListSize,
                       SQLULEN*   NumMessages)
```

The standard ODBC return codes are returned: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE, and SQL_ERROR.

Purpose

ValidateTableFromFile (ANSI application) and ValidateTableFromFileW (Unicode application) verify the metadata in the configuration file against the data structure of the target database table. See "Verification of the Bulk Load Configuration File" for more detailed information.

Parameters

hdbc

is the driver's connection handle, which is not the handle returned by SQLAllocHandle or SQLAllocConnect. To obtain the driver's connection handle, the application must then use the standard ODBC function SQLGetInfo (*ODBC Conn Handle*, *SQL_DRIVER_HDBC*).

TableName

is a null-terminated character string that specifies the name of the target database table into which the data is to be loaded.

ConfigFile

is a null-terminated character string that specifies the path (relative or absolute) and file name of the bulk configuration file.

MessageList

specifies a pointer to a buffer used to record any of the errors and warnings. MessageList must not be null.

MessageListSize

specifies the maximum number of characters that can be written to the buffer to which MessageList points. If the buffer to which MessageList points is not big enough to hold all of the messages generated by the validation process, the validation is aborted and SQL_ERROR is returned.

NumMessages

contains the number of messages that were added to the buffer. This method reports the following criteria:

- Check data types - Each column data type is checked to ensure no loss of data occurs. If a data type mismatch is detected, the driver adds an entry to the MessageList in the following format: Risk of data conversion loss: Destination *column_number* is of type *x*, and source *column_number* is of type *y*.
- Check column sizes - Each column is checked for appropriate size. If column sizes are too small in destination tables, the driver adds an entry to the MessageList in the following format: Possible Data Truncation: Destination *column_number* is of size *x* while source *column_number* is of size *y*.
- Check codepages - Each column is checked for appropriate code page alignment between the source and destination. If a mismatch occurs, the driver adds an entry to the MessageList in the following format: Destination column code page for *column_number* risks data corruption if transposed without correct character conversion from source *column_number*.
- Check Config Col Info - The destination metadata and the column metadata in the configuration file are checked for consistency of items 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. If any inconsistency is found, the driver adds an entry to the MessageList in the following format: Destination column metadata for *column_number* has column info mismatches from source *column_number*.
- Check Column Names and Mapping - The columns defined in the configuration file are compared to the destination table columns based on the order of the columns. If the number of columns in the configuration file and/or import file does not match the number of columns in the table, the driver adds an entry to the MessageList in the following format: The number of destination columns *number* does not match the number of source columns *number*.

The function returns an array of null-terminated strings in the buffer to which MessageList points with an entry for each of these checks. If the driver determines that the information in the bulk load configuration file matches the metadata of the destination table, a return code of SQL_SUCCESS is returned and the MessageList remains empty.

If the driver determines that there are minor differences in the information in the bulk load configuration file and the destination table, then SQL_SUCCESS_WITH_INFO is returned and the MessageList is populated with information on the cause of the potential problems.

If the driver determines that the information in the bulk load information file cannot successfully be loaded into the destination table, then a return code of SQL_ERROR is returned and the MessageList is populated with information on the problems and mismatches between the source and destination.

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

[Verification of the Bulk Load Configuration File](#) on page 154

LoadTableFromFile and LoadTableFromFileW

Syntax

```

SQLReturn
LoadTableFromFile (HDBC          hdbc,
                  SQLCHAR*      TableName,
                  SQLCHAR*      FileName,
                  SQLLEN        ErrorTolerance,
                  SQLLEN        WarningTolerance,
                  SQLCHAR*      ConfigFile,
                  SQLCHAR*      LogFile,
                  SQLCHAR*      DiscardFile,
                  SQLULEN       LoadStart,
                  SQLULEN       LoadCount,
                  SQLULEN       ReadBufferSize)

```

```
LoadTableFromFileW (HDBC      hdbc,  
                  SQLWCHAR* TableName,  
                  SQLWCHAR* FileName,  
                  SQLLEN     ErrorTolerance,  
                  SQLLEN     WarningTolerance,  
                  SQLWCHAR* ConfigFile,  
                  SQLWCHAR* LogFile,  
                  SQLWCHAR* DiscardFile,  
                  SQLULEN    LoadStart,  
                  SQLULEN    LoadCount,  
                  SQLULEN    ReadBufferSize)
```

The standard ODBC return codes are returned: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE, and SQL_ERROR.

Purpose

LoadTableFromFile (ANSI application) and LoadTablefromFileW (Unicode application) bulk load data from a file to a table. 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 load 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 *LoadStart* and *LoadCount* parameters can be used to control which rows are loaded when a load is restarted after a failure.

Parameters

hdbc

is the driver's connection handle, which is not the handle returned by SQLAllocHandle or SQLAllocConnect. To obtain the driver's connection handle, the application must then use the standard ODBC function SQLGetInfo (*ODBC Conn Handle, SQL_DRIVER_HDBC*).

TableName

is a null-terminated character string that specifies the name of the target database table into which the data is to be loaded.

FileName

is a null-terminated string that specifies the path (relative or absolute) and file name of the bulk data file from which the data is to be loaded. The file name must be the fully qualified path to the bulk data file.

ErrorTolerance

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.

WarningTolerance

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.

ConfigFile

is a null-terminated character string that specifies the path (relative or absolute) and file name of the bulk configuration file.

LogFile

is a null-terminated character 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. 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 *LogFile* is NULL, no log file is created.

DiscardFile is a null-terminated character 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 to be 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 *DiscardFile* is NULL, no discard file is created.

LoadStart specifies the first row to be loaded from the data file. Rows are numbered starting with 1. For example, when *LoadStart*=10, the first 9 rows of the file are skipped and the first row loaded is row 10. This parameter can be used to restart a load after a failure.

LoadCount specifies the number of rows to be loaded from the data file. The bulk load operation loads rows up to the value of *LoadCount* from the file to the database. It is valid for *LoadCount* to specify more rows than exist in the data file. The bulk load operation completes successfully when either the *LoadCount* value has been loaded or the end of the data file is reached. This parameter can be used in conjunction with *LoadStart* to restart a load after a failure.

ReadBufferSize specifies the size, in KB, of the buffer that is used to read the bulk data file for a bulk load operation. The default is 2048.

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);
}

```

DataDirect Bulk Load Statement Attributes

In addition to exporting tables with the ExportTableToFile methods, result sets can be exported to a bulk load data file through the use of two DataDirect statement attributes, SQL_BULK_EXPORT_PARAMS and SQL_BULK_EXPORT. SQL_BULK_EXPORT_PARAMS is used to configure information about where and how the data is to be exported. SQL_BULK_EXPORT begins the bulk export operation.

SQL_BULK_EXPORT

The ValuePtr argument to SQLSetStmtAttr or SQLSetStmtAttrW when the attribute argument is SQL_BULK_EXPORT is a pointer to a string that specifies the file name of the bulk load data file to which the data in the result set will be exported.

Result set export occurs when the SQL_BULK_EXPORT statement attribute is set. If using the SQL_BULK_EXPORT_PARAMS attribute to set values for the bulk export parameters, the SQL_BULK_EXPORT_PARAMS attribute must be set prior to setting the SQL_BULK_EXPORT attribute. Once set, the bulk export parameters remain set for the life of the statement. If the bulk export parameters are not set prior to setting the SQL_BULK_EXPORT attribute, the driver uses the current driver code page value, defaults EnableLogging to 1 (enabled), and defaults ErrorTolerance and WarningTolerance to -1 (infinite).

Both a bulk load data file and a bulk load configuration file are produced by this operation. The configuration file has the same base name as the bulk load data file, but with an XML extension. The configuration file is created in the same directory as the bulk load data file.

SQL_BULK_EXPORT_PARAMS

The ValuePtr argument to SQLSetStmtAttr or SQLSetStmtAttrW when the attribute argument is SQL_BULK_EXPORT_PARAMS is a pointer to a BulkExportParams structure. The definitions of the fields in the BulkExportParams structure are the same as the corresponding arguments in the ExportTableToFile and ExportTableToFileW methods except that the generation of the log file is controlled by the EnableLogging field. When EnableLogging is set to 1, the driver writes events that occur during the export to a log file. Events logged to this file are:

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

The log file is located in the same directory as the bulk load data file and has the same base name as the bulk load data file with a .log extension. When EnableLogging is set to 0, no logging takes place

If the bulk export parameters are not set prior to setting the SQL_BULK_EXPORT attribute, the driver uses the current driver code page value, defaults EnableLogging to 1 (enabled), and defaults ErrorTolerance and WarningTolerance to -1 (infinite).

The SQL_BULK_EXPORT_PARAMS structure is as follows:

```
struct BulkExportParams {
    SQLLEN  Version;                /* Must be the value 1 */
    SQLLEN  IANAAppCodePage;
    SQLLEN  EnableLogging;
    SQLLEN  ErrorTolerance;
    SQLLEN  WarningTolerance;
};
```

See also

[ExportTableToFile](#) and [ExportTableToFileW](#) on page 320

Threading

The ODBC specification mandates that all drivers must be thread-safe, that is, drivers must not fail when database requests are made on separate threads. It is a common misperception that issuing requests on separate threads always results in improved throughput. Because of network transport and database server limitations, some drivers serialize threaded requests to the server to ensure thread safety.

The ODBC 3.0 specification does not provide a method to find out how a driver services threaded requests, although this information is useful to an application. All the Progress DataDirect *for* ODBC drivers provide this information to the user through the SQLGetInfo information type 1028.

The result of calling SQLGetInfo with 1028 is a SQL_USMALLINT flag that denotes the session's thread model. A return value of 0 denotes that the session is fully thread-enabled and that all requests use the threaded model. A return value of 1 denotes that the session is restricted at the connection level. Sessions of this type are fully thread-enabled when simultaneous threaded requests are made with statement handles that do not share the same connection handle. In this model, if multiple requests are made from the same connection, the first request received by the driver is processed immediately and all subsequent requests are serialized. A return value of 2 denotes that the session is thread-impaired and all requests are serialized by the driver.

Consider the following code fragment:

```
rc = SQLGetInfo (hdbc, 1028, &ThreadModel, NULL, NULL);

If (rc == SQL_SUCCESS) {
    // driver is a DataDirect driver that can report threading information

    if (ThreadModel == 0)
        // driver is unconditionally thread-enabled; application can take advantage of
        // threading

    else if (ThreadModel == 1)
        // driver is thread-enabled when thread requests are from different connections
        // some applications can take advantage of threading

    else if (ThreadModel == 2)
        // driver is thread-impaired; application should only use threads if it reduces
```

```
    // program complexity
}
else
    // driver is not guaranteed to be thread-safe; use threading at your own risk
```

WorkAround Options

Progress DataDirect has included non-standard connection options your driver that enable you to take full advantage of packaged ODBC-enabled applications requiring non-standard or extended behavior.

To use these options, we recommend that you create a separate user data source for each application.



Your driver features the Extended Options configuration field on the Advanced tab of the driver's Setup dialog box. You can use the Extended Options field to enter undocumented connection options when instructed by Progress DataDirect Technical Support.

Alternatively, you can make the change by updating the Registry. After you create the data source,

- On Windows, using the registry editor REGEDIT, open the HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI section of the registry. Select the data source that you created.
- On UNIX/Linux/macOS, using a text editor, open the `odbc.ini` file to edit the data source that you created.

Add the string `WorkArounds=` (or `WorkArounds2=`) with a value of n (`WorkArounds=n` or `WorkArounds2=n`), where the value n is the cumulative value of all options added together. For example, if you wanted to use both `WorkArounds=1` and `WorkArounds=8`, you would enter in the data source:

```
WorkArounds=9
```

Warning: Each of these options has potential side effects related to its use. An option should only be used to address the specific problem for which it was designed. For example, `WorkArounds=2` causes the driver to report that database qualifiers are not supported, even when they are. As a result, applications that use qualifiers may not perform correctly when this option is enabled.

The following list includes both WorkArounds and WorkArounds2.

WorkArounds=1. Enabling this option causes the driver to return 1 instead of 0 if the value for SQL_CURSOR_COMMIT_BEHAVIOR or SQL_CURSOR_ROLLBACK_BEHAVIOR is 0. Statements are prepared again by the driver.

WorkArounds=2. Enabling this option causes the driver to report that database qualifiers are not supported. Some applications cannot process database qualifiers.

WorkArounds=8. Enabling this option causes the driver to return 1 instead of -1 for SQLRowCount. If an ODBC driver cannot determine the number of rows affected by an Insert, Update, or Delete statement, it may return -1 in SQLRowCount. This may cause an error in some products.

WorkArounds=16. Enabling this option causes the driver not to return an INDEX_QUALIFIER. For SQLStatistics, if an ODBC driver reports an INDEX_QUALIFIER that contains a period, some applications return a "tablename is not a valid name" error.

WorkArounds=32. Enabling this option causes the driver to re-bind columns after calling SQLExecute for prepared statements.

WorkArounds=64. Enabling this option results in a column name of *Cposition* where *position* is the ordinal position in the result set. For example, "SELECT col1, col2+col3 FROM table1" produces the column names "col1" and C2. For result columns that are expressions, SQLColAttributes/SQL_COLUMN_NAME returns an empty string. Use this option for applications that cannot process empty string column names.

WorkArounds=256. Enabling this option causes the value of SQLGetInfo/SQL_ACTIVE_CONNECTIONS to be returned as 1.

WorkArounds=512. Enabling this option prevents ROWID results. This option forces the SQLSpecialColumns function to return a unique index as returned from SQLStatistics.

WorkArounds=2048. Enabling this option causes DATABASE= instead of DB= to be returned. For some data sources, Microsoft Access performs more efficiently when the output connection string of SQLDriverConnect returns DATABASE= instead of DB=.

WorkArounds=65536. Enabling this option strips trailing zeros from decimal results, which prevents Microsoft Access from issuing an error when decimal columns containing trailing zeros are included in the unique index.

WorkArounds=131072. Enabling this option turns all occurrences of the double quote character (") into the accent grave character (`). Some applications always quote identifiers with double quotes. Double quoting can cause problems for data sources that do not return SQLGetInfo/SQL_IDENTIFIER_QUOTE_CHAR = *double_quote*.

WorkArounds=524288. Enabling this option forces the maximum precision/scale settings. The Microsoft Foundation Classes (MFC) bind all SQL_DECIMAL parameters with a fixed precision and scale, which can cause truncation errors.

WorkArounds=1048576. Enabling this option overrides the specified precision and sets the precision to 2000. Some applications incorrectly specify a precision of 0 for character types when the value will be SQL_NULL_DATA.

WorkArounds=2097152. Enabling this option overrides the specified precision and sets the precision to 2000. Some applications incorrectly specify a precision of -1 for character types.

WorkArounds=4194304. Enabling this option converts, for PowerBuilder users, all catalog function arguments to uppercase unless they are quoted.

WorkArounds=16777216. Enabling this option allows MS Access to retrieve Unicode data types as it expects the default conversion to be to SQL_C_CHAR and not SQL_C_WCHAR.

WorkArounds=33554432. Enabling this option prevents MS Access from failing when SQLError returns an extremely long error message.

WorkArounds=67108864. Enabling this option allows parameter bindings to work correctly with MSDASQL.

WorkArounds=536870912. Enabling this option allows re-binding of parameters after calling SQLExecute for prepared statements.

WorkArounds=1073741824. Enabling this option addresses the assumption by the application that ORDER BY columns do not have to be in the SELECT list. This assumption may be incorrect for data sources such as Informix.

WorkArounds2=2. Enabling this option causes the driver to ignore the ColumnSize/DecimalDigits specified by the application and use the database defaults instead. Some applications incorrectly specify the ColumnSize/DecimalDigits when binding timestamp parameters.

WorkArounds2=4. Enabling this option reverses the order in which Microsoft Access returns native types so that Access uses the most appropriate native type. Microsoft Access uses the last native type mapping, as returned by SQLGetTypeInfo, for a given SQL type.

WorkArounds2=8. Enabling this option causes the driver to add the bindoffset in the ARD to the pointers returned by SQLParamData. This is to work around an MSDASQL problem.

WorkArounds2=16. Enabling this option causes the driver to ignore calls to SQLFreeStmt(RESET_PARAMS) and only return success without taking other action. It also causes parameter validation not to use the bind offset when validating the charoctetlength buffer. This is to work around a MSDASQL problem.

WorkArounds2=24. Enabling this option allows a flat-file driver, such as dBASE, to operate properly under MSDASQL.

WorkArounds2=32. Enabling this option appends "DSN=" to a connection string if it is not already included. Microsoft Access requires "DSN" to be included in a connection string.

WorkArounds2=128. Enabling this option causes 0 to be returned by SQLGetInfo(SQL_ACTIVE_STATEMENTS). Some applications open extra connections if SQLGetInfo(SQL_ACTIVE_STATEMENTS) does not return 0.

WorkArounds2=256. Enabling this option causes the driver to return Buffer Size for Long Data on calls to SQLGetData with a buffer size of 0 on columns of SQL type SQL_LONGVARCHAR or SQL_LONGVARBINARY. Applications should always set this workaround when using MSDASQL and retrieving long data.

WorkArounds2=512. Enabling this option causes the flat-file drivers to return old literal prefixes and suffixes for date, time, and timestamp data types. Microsoft Query 2000 does not correctly handle the ODBC escapes that are currently returned as literal prefix and literal suffix.

WorkArounds2=1024. Enabling this option causes the driver to return "N" for SQLGetInfo(SQL_MULT_RESULT_SETS). ADO incorrectly interprets SQLGetInfo(SQL_MULT_RESULT_SETS) to mean that the contents of the last result set returned from a stored procedure are the output parameters for the stored procedure.

WorkArounds2=2048. Enabling this option causes the driver to accept 2.x SQL type defines as valid. ODBC 3.x applications that use the ODBC cursor library receive errors on bindings for SQL_DATE, SQL_TIME, and SQL_TIMESTAMP columns. The cursor library incorrectly rebinds these columns with the ODBC 2.x type defines.

WorkArounds2=4096. Enabling this option causes the driver to internally adjust the length of empty strings. The ODBC Driver Manager incorrectly translates lengths of empty strings when a Unicode-enabled application uses a non-Unicode driver. Use this workaround only if your application is Unicode-enabled.

WorkArounds2=8192. Enabling this option causes Microsoft Access not to pass the error -7748. Microsoft Access only asks for data as a two-byte SQL_C_WCHAR, which is an insufficient buffer size to store the UCS2 character and the null terminator; thus, the driver returns a warning, "01004 Data truncated" and returns a null character to Microsoft Access. Microsoft Access then passes error -7748.

Index

32-bit driver
 UNIX/Linux requirements [37](#)

64-bit driver
 UNIX/Linux requirements [39](#)
 Windows support [36](#)

A

about
 select failover [126](#)

Accounting Info connection option [183](#)

Action connection option [184](#)

adding connections to a connection pool [146](#)

address, IP [52](#)

Administrator
 macOS ODBC [165](#)
 macOS ODBC and tracing [165](#)
 Windows ODBC and tracing [164](#)

Administrator, Data Source
 UNIX and Linux [70](#)
 Windows [24](#)

Advanced Security tab [108](#)

Advanced tab [80](#)

AIX, See UNIX and Linux

AllowedOpenSSLVersions connection option [184](#)

alternate database servers
 about [124](#)
 guidelines for specifying [127](#)

Alternate Servers connection option [186](#)

Application Name connection option [186](#)

Application Using Threads connection option [187–188](#), [209](#)

array binding, use in bulk load operations [158](#)

Array Size connection option [188](#)

arrays of parameter values, passing [300](#)

arrays of parameters [49](#)

authentication
 SSL client [139](#)
 SSL server [138](#)
 user [133](#)

authentication, Oracle Internet Directory (OID) [135](#)

authentication, Oracle Wallet SSL [135–136](#)

Autocommit mode [302](#)

B

batch inserts, using bulk load for [158](#)

Batch Size connection option [189](#)

bound columns [299](#)

bulk
 loading to a database [152](#)

Bulk Binary Threshold connection option [190](#)

Bulk Character Threshold connection option [191](#)

Bulk Export method [150](#)

bulk load
 batch inserts [158](#)
 bulk data configuration file [154](#)
 configuration file [328](#)
 data file [328](#)
 determining the protocol [158](#)
 exporting data from a database [151](#)
 external overflow file [156](#)
 functions
 bulk errors [318](#)
 bulk export [320](#)
 bulk load [325](#)
 bulk load validation [323](#)
 utility [318](#)
 overview [149](#)
 sample application [156](#)
 statement attributes [328](#)
 validating files [153](#)
 validating metadata in the bulk load configuration file [323](#)
 verifying the configuration file [154](#)

Bulk Load method [150](#)

Bulk Options connection option [192](#)

Bulk tab [99](#)

C

Cached Cursor Limit connection option [193](#)

Cached Description Limit connection option [193](#)

caching information to improve performance [296](#)

call load, reducing [299](#)

Catalog Functions Include Synonyms connection option [194](#)

catalog functions, using [296](#)

Catalog Options connection option [195](#)

changes to behavior for release 8.0.0 [14](#)

changes to behavior for release 8.0.1 [14](#)

changes to behavior for release 8.0.2 [14](#)

character encoding [285](#)

character set conversions [156](#)

characters, unexpected [122](#)

cipher suite, encryption
 SSL v3 encryption cipher suite [309](#)
 TLS v1 encryption cipher suite [309](#)
 when driver cannot negotiate SSL v3 or TLS v1 [309](#)

ClearPool and ClearAllPools methods [146](#)

client code page, See code pages

Client Host Name connection option [195](#)

- Client ID connection option [196](#)
- client information
 - about [132](#)
 - how databases store [132](#)
 - location used for storing [132](#)
 - storing [132](#)
- client load balancing, about [127](#)
- Client Monitoring tab [106](#)
- Client User connection option [197](#)
- Close() method
 - effect on the connection string [146](#)
- code pages
 - IANAAppCodePage attribute [267](#)
 - IANAAppCodePage values [267](#)
- comma-separated value (CSV) format file
 - character set conversions [156](#)
 - use in bulk load [154](#)
- configuration file
 - bulk data [154](#)
- configuring a data source
 - macOS [28](#)
 - UNIX and Linux [26](#), [58](#)
 - using a GUI [75](#)
- connecting
 - proxy server [119](#), [121](#)
- connecting via connection string [113](#)
- connection attribute
 - AccountingInfo [183](#)
 - Action [184](#)
 - AllowedOpenSSLVersions [184](#)
 - AlternateServers [186](#)
 - ApplicationName [186](#)
 - ApplicationUsingThreads [187–188](#), [209](#)
 - ArraySize [188](#)
 - BulkBinaryThreshold [190](#)
 - BulkCharacterThreshold [191](#)
 - BulkLoadBatchSize [189](#)
 - BulkLoadFieldDelimiter [221](#)
 - BulkLoadOptions [192](#)
 - BulkLoadRecordDelimiter [246](#)
 - CachedCursorLimit [193](#)
 - CachedDescriptionLimit [193](#)
 - CatalogIncludesSynonyms [194](#)
 - CatalogOptions [195](#)
 - ClientHostName [195](#)
 - ClientID [196](#)
 - ClientUser [197](#)
 - ConnectionRetryCount [199](#)
 - ConnectionRetryDelay [200](#)
 - CredentialsWalletEntry [201](#)
 - CredentialsWalletPassword [260](#)
 - CredentialsWalletPath [202](#)
 - CryptoLibName [204](#)
 - CryptoProtocolVersion [203](#)
 - DataIntegrityLevel [205](#)
 - DataIntegrityTypes [206](#)
 - connection attribute (*continued*)
 - DataSourceName [207](#)
 - DefaultLongDataBuffLen [207](#)
 - DescribeAtPrepare [208](#)
 - Description [208](#)
 - EnableBulkLoad [210](#)
 - EnableDescribeParam [213](#)
 - EnableNcharSupport [211](#)
 - EnableScrollableCursors [212](#)
 - EnableServerResultCache [212](#)
 - EnableStaticCursorsForLongData [214](#)
 - EnableTimestampwithTimezone [214](#)
 - EncryptionLevel [215](#)
 - EncryptionMethod [216](#)
 - EncryptionTypes [217](#)
 - FailoverGranularity [218](#)
 - FailoverMode [219](#)
 - FailoverPreconnect [220](#)
 - FetchTSWTZasTimestamp [220](#)
 - GSSClient [222](#)
 - HostName [222](#)
 - HostNameInCertificate [223](#)
 - IANAAppCodePage [224](#)
 - ImpersonateUser [225](#)
 - InitializationString [226](#)
 - KeyPassword [226](#)
 - Keystore [227](#)
 - KeystorePassword [228](#)
 - LDAPDistinguishedName [228](#)
 - LoadBalanceTimeout [230](#)
 - LoadBalancing [229](#)
 - LOBPrefetchSize [231](#)
 - LocalTimezoneOffset [231](#)
 - LockTimeout [232](#)
 - LoginTimeout [233](#)
 - LogonID [259](#)
 - MaxPoolSize [234](#)
 - MinPoolSize [234](#)
 - Module [235](#)
 - Password [236](#)
 - Pooling [198](#)
 - PortNumber [237](#)
 - PRNGSeedFile [242](#)
 - PRNGSeedSource [243](#)
 - ProcedureRetResults [244](#)
 - ProgramID [245](#)
 - QueryTimeout [246](#)
 - ReportCodepageConversionErrors [247](#)
 - ReportRecycleBin [248](#)
 - SDUSize [248](#)
 - ServerName [249](#)
 - ServerType [250](#)
 - ServiceName [251](#)
 - SID [252](#)
 - SSLLibName [253](#)
 - SupportBinaryXML [254](#)

- connection attribute (*continued*)
 - TimestampEscapeMapping [255](#)
 - TNSNamesFile [256](#)
 - Truststore [257](#)
 - TruststorePassword [258](#)
 - UseCurrentSchema [258](#)
 - ValidateServerCertificate [259](#)
 - WireProtocolMode [261](#)
- connection attributes
 - KeepAlive [254](#)
 - ProxyHost [237](#)
 - ProxyMode [238](#)
 - ProxyPassword [239](#)
 - ProxyPort [240](#)
 - ProxyUser [241](#)
- connection failover
 - about [123–124](#)
 - connection retry and [124](#)
 - load balancing and [127](#)
- connection handles [302](#)
- connection issues [170](#)
- connection options
 - Accounting Info [183, 254](#)
 - Action [184](#)
 - AllowedOpenSSLVersions [184](#)
 - Alternate Servers [186](#)
 - Application Name [186](#)
 - Application Using Threads [187–188, 209](#)
 - Array Size [188](#)
 - Batch Size [189](#)
 - Bulk Binary Threshold [190](#)
 - Bulk Character Threshold [191](#)
 - Bulk Options [192](#)
 - Cached Cursor Limit [193](#)
 - Cached Description Limit [193](#)
 - Catalog Functions Include Synonyms [194](#)
 - Catalog Options [195](#)
 - Client Host Name [195](#)
 - Client ID [196](#)
 - Client User [197](#)
 - Connection Pooling [198](#)
 - Connection Retry Count [199](#)
 - Connection Retry Delay [200](#)
 - CredentialsWalletEntry [201](#)
 - CredentialsWalletPassword [260](#)
 - CredentialsWalletPath [202](#)
 - Crypto Protocol Version [203](#)
 - CryptoLibName [204](#)
 - Data Integrity Level [205](#)
 - Data Integrity Types [206](#)
 - Data Source Name [207](#)
 - Default Buffer Size for Long/LOB Columns (in Kb) [207](#)
 - Describe at Prepare [208](#)
 - Description [208](#)
 - Enable Bulk Load [210](#)
- connection options (*continued*)
 - Enable N-CHAR Support [211](#)
 - Enable Scrollable Cursors [212](#)
 - Enable Server Result Cache [212](#)
 - Enable SQLDescribeParam [213](#)
 - Enable Static Cursors for Long Data [214](#)
 - Enable Timestamp with Timezone [214](#)
 - Encryption Level [215](#)
 - Encryption Method [216](#)
 - Encryption Types [217](#)
 - Failover Granularity [218](#)
 - Failover Mode [219](#)
 - Failover Preconnect [220](#)
 - Fetch TSWTZ as Timestamp [220](#)
 - Field Delimiter [221](#)
 - for Bulk Load [159](#)
 - for DataDirect Bulk Load [157](#)
 - for failover [128](#)
 - for security [141](#)
 - GSS Client Library [222](#)
 - Host [222](#)
 - Host Name In Certificate [223](#)
 - IANAAppCodePage [224](#)
 - Impersonate User [225](#)
 - Initialization String [226](#)
 - Key Password [226](#)
 - Key Store [227](#)
 - Key Store Password [228](#)
 - LDAP Distinguished Name [228](#)
 - Load Balancing [229](#)
 - LoadBalance Timeout [230](#)
 - LOB Prefetch Size [231](#)
 - Local Timezone Offset [231](#)
 - Lock Timeout [232](#)
 - Login Timeout [233](#)
 - Max Pool Size [234](#)
 - Min Pool Size [234](#)
 - Module [235](#)
 - Password [236](#)
 - Port Number [237](#)
 - PRNGSeedFile [242](#)
 - PRNGSeedSource [243](#)
 - Procedure Returns Results [244](#)
 - Program ID [245](#)
 - Proxy Host [237](#)
 - Proxy Mode [238](#)
 - Proxy Password [239](#)
 - Proxy Port [240](#)
 - Proxy User [241](#)
 - Query Timeout [246](#)
 - Record Delimiter [246](#)
 - Report Codepage Conversion Errors [247](#)
 - Report Recycle Bin [248](#)
 - required [75](#)
 - SDU Size [248](#)
 - Server Name [249](#)

- connection options (*continued*)
 - Server Process Type [250](#)
 - Service Name [251](#)
 - SID [252](#)
 - SSLLibName [253](#)
 - TCP Keep Alive [254](#)
 - Timestamp Escape Mapping [255](#)
 - TNSNames File [256](#)
 - Trust Store [257](#)
 - Trust Store Password [258](#)
 - Use Current Schema for SQLProcedures [258](#)
 - User Name [259](#)
 - Validate Server Certificate [259](#)
 - Wire Protocol Mode [261](#)
 - connection pool
 - adding connections [146](#)
 - ClearPool and ClearAllPools methods [146](#)
 - creating [146](#)
 - dead connections [147](#)
 - removing connections [146](#)
 - returning connections to [146](#)
 - when a physical connection to a server is lost [147](#)
 - connection pooling [145](#)
 - Connection Pooling
 - connection properties
 - for connection pooling [148](#)
 - Connection Pooling connection option [198](#)
 - Connection Reset [198](#)
 - connection retry
 - about [124](#)
 - Connection Retry Count connection option [199](#)
 - Connection Retry Delay connection option [200](#)
 - connection retry, about [128](#)
 - connection string attributes
 - Oracle Wire Protocol [175](#)
 - connection string options
 - effect of Close and Dispose methods [146](#)
 - connections
 - optimizing [302](#)
 - connections supported [52](#)
 - contacting Technical Support [20](#)
 - Credentials Wallet Entry connection option [201](#)
 - Credentials Wallet Path connection option [202](#)
 - Crypto Protocol Version connection option [203](#)
 - CryptoLibName connection option [204](#)
 - CSV, See comma-separated value
 - cursor library, performance implications of using [301](#)
- D**
- data encryption [85](#), [137](#)
 - data integrity [137](#), [141](#)
 - Data Integrity Level connection option [205](#)
 - Data Integrity Types connection option [206](#)
 - data retrieval, optimizing [298](#)
 - data source
 - configuring
 - macOS [28](#), [66](#)
 - UNIX and Linux [25](#), [58](#)
 - Windows [24](#)
 - connecting via connection string [113](#)
 - connecting via logon dialog box [113](#)
 - Data Source Administrator
 - macOS [70](#)
 - Windows [24](#)
 - Data Source Name connection option [207](#)
 - data types
 - choosing [300](#)
 - retrieving information [48](#)
 - database connections, testing [168](#)
 - database versions supported [33](#)
 - DataDirect Bulk Load
 - about [149](#)
 - character set conversions [156](#)
 - external overflow file [156](#)
 - date and time functions [280](#)
 - ddtestlib tool
 - macOS [65](#)
 - dead connections in a connection pool [147](#)
 - dedicated bulk protocol [158](#)
 - Default Buffer Size for Long/LOB Columns (in Kb)
 - connection option [207](#)
 - Describe at Prepare connection option [208](#)
 - Description connection option [208](#)
 - determining optimal set of columns [303](#)
 - diagnostic tools [163](#)
 - dirty reads [306](#)
 - Dispose() method, effect on the connection string [146](#)
 - Distributed Transaction Coordinator [53](#)
 - distributed transactions [53](#), [303](#)
 - DNS-less connections [63](#)
 - documentation, about [19](#)
 - double-byte character sets in UNIX and Linux [285](#)
 - driver
 - API and scalar functions [273](#)
 - code page values [267](#)
 - connections supported [52](#)
 - ODBC compliance [34](#)
 - optimizing application performance [295](#)
 - statements supported [52](#)
 - supported features [51](#)
 - testing the connection [25](#)
 - the driver (overview) [55](#)
 - threading [331](#)
 - using DataDirect bulk load [317](#)
 - version string information [43](#)
 - Driver Manager
 - macOS [28](#), [65](#)
 - support [73](#)
 - driver requirements [34](#)

E

Enable Bulk Load connection option [210](#)
 Enable N-CHAR Support connection option [211](#)
 Enable Scrollable Cursors connection option [212](#)
 Enable Server Result Cache connection option [212](#)
 Enable SQLDescribeParam connection option [213](#)
 Enable Static Cursors for Long Data connection option [214](#)
 Enable Timestamp with Timezone connection option [214](#)
 enabling tracing [164](#)
 encryption

- data [137](#)
- Oracle [137](#)
- SSL [138](#)

 encryption cipher suites [309](#)
 Encryption Level connection option [215](#)
 Encryption Method connection option [216](#)
 Encryption Types connection option [217](#)
 environment

- variables [56](#)

 environment variable, library path [171](#)
 environments supported [35](#)
 error messages

- general [169](#)
- macOS [169](#)
- UNIX and Linux [169](#)
- Windows [169](#)

 example

- bulk load sample application [156](#)
- character set conversions [156](#)

 example application [168](#)
 executing SQL [168](#)
 exporting data from a database [151](#)
 exporting result sets to a bulk load data file [328](#)
 ExportTableToFile [320](#)
 ExportTableToFileW [320](#)
 extended connection failover

- about [125](#)

 Extended Options [80](#), [333](#)
 external overflow file [156](#)

F

Failover

- configuring [94](#)

 Failover Granularity connection option [218](#)
 Failover Mode connection option [219](#)
 Failover Preconnect connection option [220](#)
 Failover Tab [94](#)
 Fetch TSWTZ as Timestamp connection option [220](#)
 Field Delimiter connection option [221](#)
 file data sources [64](#)
 file names

- Windows [37](#)

functionality

- changes [14](#)
- new [14](#)

 functions, ODBC

- DataDirect functions for bulk operations [317](#)
- selecting for performance [300](#)

G

General Tab [75](#)
 GetBulkDiagRec [318](#)
 GetBulkDiagRecW [318](#)
 GSS Client Library connection option [222](#)
 guidelines for primary and alternate servers [127](#)

H

handles

- connection [302](#)
- statement [302](#)

 Host connection option [222](#)
 Host Name In Certificate connection option [223](#)
 HP-UX, See UNIX and Linux

I

IANAAppCodePage

- connection option values [267](#)

 IANAAppCodePage connection option [224](#)
 Impersonate User connection option [225](#)
 improving

- database performance [263](#)
- index performance [263](#)
- join performance [266](#)
- ODBC application performance [173](#), [295](#)
- record selection performance [264](#)

 indexes

- deciding which to create [265](#)
- improving performance [263](#)
- overview [264](#)

 indexing multiple fields [265](#)
 Initialization String connection option [226](#)
 integrity [141](#)
 internationalization [283](#)
 interoperability issues [172](#)
 iODBC Demo [168](#)
 iODBC Driver Manager [28](#), [65](#)
 IP addresses [52](#)
 IPv4 [52](#)
 IPv6 [52](#)
 isolation levels

- about [306](#)
- read committed [306](#)
- read uncommitted [306](#)
- repeatable read [306](#)

isolation levels (*continued*)

serializable [306](#)

isolation levels and data consistency

compared [306](#)

dirty reads [306](#)

non-repeatable reads [306](#)

phantom reads [306](#)

ivtestlib tool [167](#)

K

KeepAlive

connection attribute [254](#)

Kerberos authentication

kinit command [134](#)

Ticket Granting Ticket (TGT) [134](#)

Key Password connection option [226](#)

Key Store connection option [227](#)

Key Store Password connection option [228](#)

kinit command, Kerberos authentication [134](#)

L

LDAP [119](#)

LDAP Distinguished Name connection option [228](#)

library path environment variable [171](#)

Linux, See UNIX and Linux

listener.ora [251](#)

Load Balancing connection option [229](#)

load balancing, about [127](#)

LoadBalance Timeout connection option [230](#)

LoadTableFromFile [325](#)

LoadTableFromFileW [325](#)

LOB Prefetch Size connection option [231](#)

Local Timezone Offset connection option [231](#)

locale [284](#)

localization [283](#)

location used for storing client information for a connection [132](#)

Lock Timeout connection option [232](#)

locking [305](#)

locking levels [49](#)

locking modes and levels [307](#)

Login Timeout connection option [233](#)

long data, retrieving [298](#)

lost connections

extended connection failover [125](#)

recovering work in progress [126](#)

select failover [126](#)

M

macOS

configuring a data source [28](#)

macOS (*continued*)

configuring through the system information (odbc.ini) file [66](#)

Data Source Administrator [70](#)

data source configuration [28](#), [66](#)

driver manager [28](#), [65](#)

driver names [43](#)

environment

ddtestlib tool [65](#)

introduction [65](#)

ODBCINST [75](#)

system information file (.odbc.ini) [66](#)

variables [74](#)

error messages [169](#)

operating systems, supported [42](#)

system information (odbc.ini) file, configuring through [66](#)

system requirements [42](#)

macOS iODBC Administrator and tracing [165](#)

managing connections [302](#)

materialized views [54](#)

Max Pool Size connection option [234](#)

MIBenum value [267](#)

Min Pool Size connection option [234](#)

Module connection option [235](#)

MTS support [53](#)

N

New Features and Enhancements for Release 8.0.0 [14](#)

New Features and Enhancements for Release 8.0.1 [14](#)

New Features and Enhancements for Release 8.0.2 [14](#)

non-repeatable reads [306](#)

NTLM authentication [133](#)

numeric functions [279](#)

O

ODBC

API functions [273](#)

call load, reducing [299](#)

designing for performance [295](#)

functions, selecting for performance [300](#)

how the architecture works [32](#)

scalar functions [276](#)

specification [34](#)

ODBC application performance design [173](#)

ODBC conformance [34](#)

ODBC Test [168](#)

ODBC Trace [163](#)

odbc.ini

encoding [292](#)

macOS [68](#)

sample [60](#), [68](#)

UNIX/Linux [60](#)

- odbc.ini (system information) file
 - configuring with
 - macOS [66](#)
- odbc.ini file
 - controlling tracing with [166](#)
- odbcinst.ini
 - encoding [292](#)
- odbcinst.ini file
 - sample file [63](#)
 - using [63](#)
- OEM to ANSI translation [160](#)
- Open Database Connectivity, See ODBC specification
- OpenSSL cipher suites
 - driver support [138](#)
- OpenSSL cipher suites, mapping [309](#)
- OpenSSL Library
 - designating [139](#)
- operating systems
 - macOS [42](#)
 - UNIX and Linux
 - supported [39](#)
 - UNIX/Linux
 - 32-bit [37](#)
 - Windows
 - supported [35–36](#)
- optimization, performance [295](#)
- Oracle
 - encryption
 - Oracle Advanced Security [137](#)
 - encryption, configuring [140](#)
 - Wallet [140](#)
- Oracle Advanced Security
 - data integrity [137](#)
 - encryption [137](#)
- Oracle Real Application Clusters (RAC) [52](#)
- Oracle System Identifier, specifying [252](#)
- Oracle Wire Protocol driver
 - connection string attributes [175](#)
 - materialized views [54](#)
 - stored procedures [53](#)
- OS authentication [53](#), [135](#)
- overflow files for bulk load operations [156](#)

P

- parameter arrays [49](#)
- parameter values, passing arrays of [300](#)
- Password connection option [236](#)
- performance [173](#)
- performance considerations [115](#)
- performance optimization
 - avoiding catalog functions [296](#)
 - avoiding search patterns [297](#)
 - commits in transactions [302](#)
 - managing connections [302](#)
 - overview [295](#)

- performance optimization (*continued*)
 - reducing the size of retrieved data [298](#)
 - retrieving long data [298](#)
 - using a dummy query [297](#)
 - using bound columns [299](#)
- Performance tab [90](#)
- performance, improving
 - database using indexes [263](#)
 - index [263](#)
 - join [266](#)
 - record selection [264](#)
- phantom reads [306](#)
- Pooling tab [97](#)
- Port Number connection option [237](#)
- positioned updates and deletes [303](#)
- primary server, specifying guidelines for [127](#)
- PRNGSeedFile connection option [242](#)
- PRNGSeedSource connection option [243](#)
- Procedure Returns Results connection option [244](#)
- Program ID connection option [245](#)
- Proxy Host connection option [237](#)
- Proxy Mode connection option [238](#)
- Proxy Password connection option [239](#)
- Proxy Port connection option [240](#)
- proxy server, connecting through [119](#), [121](#)
- Proxy tab [111](#)
- Proxy User connection option [241](#)

Q

- Query Timeout connection option [246](#)
- quick start [23](#)

R

- read committed [306](#)
- read uncommitted [306](#)
- Real Application Clusters (RAC) [52](#)
- Record Delimiter connection option [246](#)
- removing connections from a connection pool [146](#)
- repeatable read [306](#)
- Report Codepage Conversion Errors connection option [247](#)
- Report Recycle Bin connection option [248](#)
- retrieving data type information [48](#)
- retrieving data, optimizing [298](#)

S

- scalar functions, ODBC [276](#)
- scrollable cursors [301](#)
- SDU Size connection option [248](#)
- search patterns, avoiding [297](#)
- Secure Sockets Layer, See SSL

- security
 - configuring [85](#)
 - encryption and data integrity [141](#)
 - Security Tab [85](#)
 - select failover [126](#)
 - serializable isolation level [306](#)
 - Server Name connection option [249](#)
 - Server Process Type connection option [250](#)
 - service class [115](#)
 - Service Name connection option [251](#)
 - setup issues [170](#)
 - SID [252](#)
 - SID connection option [252](#)
 - Solaris, See UNIX and Linux
 - SQL
 - executing [168](#)
 - SQL support [53](#)
 - SQLExecDirect, advantages of using [300](#)
 - SQLFetch, disadvantage of using [299](#)
 - SQLSpecialColumns, determining optimal set of columns [303](#)
 - SSL
 - client authentication [139](#)
 - configuring
 - Oracle Wallet [140](#)
 - encryption [138](#)
 - encryption cipher suites [309](#)
 - server authentication [138](#)
 - SSLLibName connection option [253](#)
 - standards, ODBC specification compliance [34](#)
 - statement attributes for DataDirect bulk load operations [328](#)
 - statement handles, using to manage SQL statements [302](#)
 - statements supported [52](#)
 - static cursors [301](#)
 - stored procedures
 - Oracle Wire Protocol [53](#)
 - stored results
 - Oracle Wire Protocol [53](#)
 - storing client information [132](#)
 - string functions [277](#)
 - Support Binary XML connection option [254](#)
 - supported database versions [33](#)
 - system functions [282](#)
 - system identifier [252](#)
 - system information file (.odbc.ini)
 - macOS [66](#)
- T**
- TCP Keep Alive
 - connection option [254](#)
 - Technical Support, contacting [20](#)
 - test connect
 - UNIX and Linux [27](#)
 - testing database connections [168](#)
 - testing the connection
 - macOS [29](#)
 - threading, overview [331](#)
 - Ticket Granting Ticket (TGT), Kerberos authentication [134](#)
 - time functions [280](#)
 - Timestamp Escape Mapping connection option [255](#)
 - TLS v1 encryption cipher suite [309](#)
 - TNSNames File connection option [256](#)
 - tools
 - diagnostic [163](#)
 - other [168](#)
 - trace log [163](#)
 - tracing
 - creating a trace log [163](#)
 - enabling with macOS iODBC Administrator [165](#)
 - enabling with system information file [166](#)
 - enabling with Windows ODBC Administrator [164](#)
 - macOS
 - iODBC [72](#)
 - transactions, managing commits [302](#)
 - troubleshooting [163](#), [170](#)
 - Trust Store connection option [257](#)
 - Trust Store Password connection option [258](#)
- U**
- UCS-2 [285](#)
 - undocumented connection options [80](#)
 - unexpected characters [119](#), [122](#)
 - Unicode
 - character encoding [285](#)
 - ODBC drivers [287](#)
 - support in databases [286](#)
 - support in ODBC [286](#)
 - Unicode support
 - macOS [74](#)
 - UNIX and Linux
 - configuring a data source [26](#), [58](#)
 - data source configuration [25](#), [58](#)
 - double-byte character sets [285](#)
 - driver names [41](#)
 - environment
 - DD_INSTALLDIR [58](#)
 - introduction [56](#)
 - library search path [56](#)
 - ODBCINI [57](#)
 - ODBCINST [57](#)
 - system information file (.odbc.ini) [58](#)
 - error messages [169](#)
 - system requirements [37](#)
 - UNIX/Linux requirements
 - 32-bit driver [37](#)
 - 64-bit driver [39](#)
 - updates, optimizing [302](#)

Use Current Schema for SQLProcedures connection option [258](#)
user authentication [133](#)
User Name connection option [259](#)
using
 bulk load for batch inserts [158](#)
 the driver (overview) [55](#)
UTF-16 [285](#)
UTF-16 Applications [65](#)
UTF-32 Applications [74](#)
UTF-8 [285](#)

V

Validate Server Certificate connection option [259](#)
ValidateTableFromFile [323](#)
ValidateTableFromFileW [323](#)
validating bulk load files [153](#)
validating metadata in the bulk load configuration file [323](#)
verifying the bulk load configuration file [154](#)
version string information [43](#)

views
 materialized [54](#)

W

Wallet Password connection option [260](#)
What's new? [14](#)
Windows
 configuring a data source using a GUI [75](#)
 Data Source Administrator [24](#)
 data source configuration [24](#)
 driver names [37](#)
 error messages [169](#)
 system requirements [35](#)
Windows ODBC Administrator and tracing [164](#)
Wire Protocol Mode connection option [261](#)
WorkAround options for ODBC drivers [333](#)

X

XA Protocol [53](#)
XML data type [46](#)

