Progress | DataDirect

Progress® DataDirect Connect® Series

for ODBC

User's Guide

Release 6.0 Service Pack 2
October 2009

© 2009 Progress Software Corporation. All rights reserved. Printed in the U.S.A.

DataDirect, DataDirect Connect, DataDirect Connect64, DataDirect Spy, DataDirect Test, DataDirect XML Converters, DataDirect XQuery, OpenAccess, SequeLink, Stylus Studio, and SupportLink are trademarks or registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the United States and other countries. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. MySQL and MySQL Enterprise are registered trademarks of MySQL AB in the United States, the European Union and other countries.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective companies.

DataDirect products for the Microsoft SQL Server database:

These products contain a licensed implementation of the Microsoft TDS Protocol.

DataDirect Connect for ODBC, DataDirect Connect64 for ODBC, and DataDirect SequeLink include:

ICU Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

Software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http:/www.openssl.org/). Copyright © 1998-2006 The OpenSSL Project. All rights reserved. And Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

DataDirect SequeLink includes:

Portions created by Eric Young are Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All Rights Reserved. OpenLDAP, Copyright © 1999-2003 The OpenLDAP Foundation, Redwood City, California, US. All rights reserved.

DataDirect OpenAccess SDK client for ODBC, DataDirect OpenAccess SDK client for ADO, DataDirect Open Access SDK client for JDBC, and DataDirect OpenAccess SDK server include: DataDirect SequeLink.

No part of this publication, with the exception of the software product user documentation contained in electronic format, may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without prior written consent of DataDirect Technologies.

Licensees may duplicate the software product user documentation contained on a CD-ROM or DVD, but only to the extent necessary to support the users authorized access to the software under the license agreement. Any reproduction of the documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

Table of Contents

List of Tables 21

Preface 25

Product Platforms 25

Product Matrix 27

What's New in This Release? 28

Using this Book 29

Conventions Used in This Book 30
Typographical Conventions 30
Environment-Specific Information 32

About the Product Documentation 32
HTML Version 32
PDF Version 34

Contacting Technical Support 35

Part 1: Getting Started

1 Quick Start Connect 41

Configuring and Connecting on Windows 42
Configuring a Data Source Through the Administrator . . 42
Testing the Connection 43

Configuring and Connecting on UNIX and Linux 44
Environment Configuration 45
Test Loading the Driver 46
Configuring a Data Source Through the Administrator . . 46
Testing the Connection 48

Using the Performance Wizard 49
Starting the Wizard 50
Tuning Performance Using the Wizard 50

2 Using The Product 55

What Is ODBC? 55
How Does It Work? 56
Why Do Application Developers Need ODBC? 57

About the Product 58

Environment-Specific Information 59
For Windows Users 59
For UNIX and Linux Users 62

Using IP Addresses 70

Binding Parameter Markers 72

Driver Threading Information 73

Version String Information 74

Retrieving Data Type Information 77

Persisting a Result Set as an XML Data File 79
Using the Windows XML Persistence Demo Tool 80
Using the UNIX/Linux XML Persistence Demo Tool 83

Translators 83

3 Advanced Features 85

Using Failover 85
Connection Failover 87
Extended Connection Failover 89
Select Connection Failover 92
Guidelines for Primary and Alternate Servers 94
Using Client Load Balancing 95
Using Connection Retry 96
Summary of Failover-Related Options 97

Using Security 101
Authentication 102
Data Encryption Across the Network 103
SSL Encryption 104
Summary of Security-Related Options 108

Using DataDirect Connection Pooling 109
Creating a Connection Pool 110
Adding Connections to a Pool 110
Removing Connections from a Pool 111
Handling Dead Connections in a Pool 112
Connection Pool Statistics 113
Summary of Pooling-Related Options 114

Using DataDirect Bulk Load 115
Bulk Export and Load Methods 117
Exporting Data from a Database 118
Bulk Loading to a Database 121
The Bulk Load Configuration File 123
Sample Application 127
Determining the Bulk Load Protocol 128
Character Set Conversions 128
External Overflow Files 129
Summary of Bulk Load Related Options 130

4 Configuring the Product on UNIX/Linux 131

Environment Variables 131
Library Search Path 132
ODBCINI 133
ODBCINST 134
DD_INSTALLDIR 135

The Test Loading Tool 136

Data Source Configuration 137
Configuration Through the Administrator 138
Configuration Through the odbc.ini File 141

The demoodbc Application 160
The example Application 160
DSN-less Connections 161
File Data Sources 167
UTF-16 Applications on UNIX and Linux 168

Part 2: The 32-Bit/64-Bit Drivers

5 The DB2 Wire Protocol Driver 173

Driver Requirements 174
Binding 174
Creating DB2 Packages Using List Files 176
Advanced Features 177
Kerberos Authentication Requirements 177
Workload Manager (WLM) 178
Configuring and Connecting to Data Sources 181
Data Source Configuration in the UNIX odbc.ini File. . . . 181
Data Source Configuration through a GUI 182
Using a Connection String 205
Using a Logon Dialog Box 206
Connection Option Descriptions 208
Performance Considerations 250
IBM to IANA Code Page Values 253
Data Types. 253
Using the XML Data Type. 254
Unicode Support. 255
Unexpected Characters 255
XQuery Expressions 258
Stored Procedure Support 258

Using DataDirect Bulk Load on DB2 258
Using Connection Failover 259
Persisting a Result Set as an XML Data File 259
Isolation and Lock Levels Supported 259
ODBC Conformance Level 260
Number of Connections and Statements Supported. 260
Using Arrays of Parameters. 260

6 The Greenplum Wire Protocol Driver 263

Driver Requirements 263
Advanced Features 264
Configuring and Connecting to Data Sources 264
Data Source Configuration in the UNIX odbc.ini File 265
Data Source Configuration through a GUI 266
Using a Connection String. 274
Using a Logon Dialog Box 275
Connection Option Descriptions. 276
Performance Considerations. 299
Data Types 300
Using the XML Data Type 302
Unicode Support 302
Stored Procedure Results 302
Configuring Failover 305
Persisting a Result Set as an XML Data File 305
Isolation and Lock Levels Supported 305
ODBC Conformance Level 306
Number of Connections and Statements Supported. 306
Using Arrays of Parameters. 307

7 The Informix Wire Protocol Driver 309

Driver Requirements 309

Advanced Features 310

Configuring and Connecting to Data Sources 310
- Data Source Configuration in the UNIX odbc.ini File. . . . 311
- Data Source Configuration through a GUI 312
- Using a Connection String 317
- Using a Logon Dialog Box 319

Connection Option Descriptions 320

Performance Considerations 331

Data Types. 332

MTS Support 334

Configuring Connection Failover 334

Persisting a Result Set as an XML Data File 334

Isolation and Lock Levels Supported 335

ODBC Conformance Level 335

Number of Connections and Statements Supported 335

8 The MySQL Wire Protocol Driver 337

Driver Requirements 338

Advanced Features 338

Configuring and Connecting to Data Sources 338
- Data Source Configuration in the UNIX odbc.ini File. . . . 339
- Data Source Configuration through a GUI 340
- Using a Connection String 350
- Using a Logon Dialog Box 351

Connection Options Descriptions 352

Performance Considerations 378

Data Types. 380

Unicode Support 381
Configuring Connection Failover 382
Persisting a Result Set as an XML Data File 382
Isolation and Lock Levels Supported 382
ODBC Conformance Level 383
Number of Connections and Statements Supported. 383

9 The Oracle Wire Protocol Driver 385

Driver Requirements 385
Advanced Features 386
Kerberos Authentication Requirements. 386
Configuring and Connecting to Data Sources 387
Data Source Configuration in the UNIX odbc.ini File 388
Data Source Configuration through a GUI 389
Using a Connection String. 412
Using a Logon Dialog Box 413
Connection Option Descriptions. 415
Performance Considerations. 461
Data Types 466
XMLType 468
Unicode Support 470
Unexpected Characters 470
MTS Support 473
OS Authentication 473
Support for Oracle RAC. 474
Support of Materialized Views 474
Stored Procedure Results 475
SSL and Oracle 476

Using DataDirect Bulk Load on Oracle 476
Limitations 477

Configuring Connection Failover 477

Persisting a Result Set as an XML Data File 477

Isolation and Lock Levels Supported 478

ODBC Conformance Level 478

Number of Connections and Statements Supported 479

Using Arrays of Parameters 479

10 The PostgreSQL Wire Protocol Driver 481

Driver Requirements 481

Advanced Features 482

Configuring and Connecting to Data Sources 482
Data Source Configuration in the UNIX odbc.ini File. . . . 483
Data Source Configuration through a GUI 484
Using a Connection String 494
Using a Logon Dialog Box 495

Connection Option Descriptions 496

Performance Considerations 524

Data Types. 526
Using the XML Data Type. 527

Unicode Support. 528

Stored Procedure Results 528

Configuring Failover 531

Persisting a Result Set as an XML Data File 531

Isolation and Lock Levels Supported 531

ODBC Conformance Level 532

Number of Connections and Statements Supported 532

Using Arrays of Parameters 533

11 The SQL Server Native Wire Protocol Driver 535

Driver Requirements 535

Advanced Features 536
Kerberos Authentication Requirements. 536

Configuring and Connecting to Data Sources 537
Data Source Configuration in the UNIX odbc.ini File 538
Data Source Configuration through a GUI 539
Using a Connection String. 559
Using a Logon dialog Box 560

Connection Option Descriptions. 562

Performance Considerations. 597

Data Types 599
Using the XML Data Type 601

Unicode Support 602

Using DataDirect Bulk Load on SQL Server 603

Configuring Connection Failover 603

Persisting a Result Set as an XML Data File 603

Isolation and Lock Levels Supported 604
Using The Snapshot Isolation Level 604

ODBC Conformance Level 605

Number of Connections and Statements Supported. 605

Using Arrays of Parameters. 606

12 The Sybase Wire Protocol Driver 607

Driver Requirements 607

Advanced Features 608
Kerberos Authentication Requirements. 608

Configuring and Connecting to Data Sources 609
Data Source Configuration in the UNIX odbc.ini File 610
Data Source Configuration through a GUI 611

Using a Connection String 635
Using a Logon Dialog Box 637

Connection Option Descriptions 638

Performance Considerations 680

Data Types. 684

Unicode Support. 685

Unexpected Characters 686

MTS Support 689

NULL Values 690

Using DataDirect Bulk Load on Sybase 691
Bulk Copy Operations and Transactions 691
Performance Considerations 692

Configuring Connection Failover 692

Persisting a Result Set as an XML Data File 693

Isolation and Lock Levels Supported. 693

ODBC Conformance Level 693

Number of Connections and Statements Supported 694

Using Arrays of Parameters 694

13 The Oracle Driver. 695

Driver Requirements 695
Windows 696
UNIX and Linux. 696

Advanced Features. 698

Configuring and Connecting to Data Sources 699
Data Source Configuration in the UNIX odbc.ini File. . . . 699
Data Source Configuration through a GUI 700
Using a Connection String 710
Using a Logon Dialog Box 711

Connection Option Descriptions 712
Performance Considerations 733
Data Types 737
XMLType 738
Unicode Support 740
Unexpected Characters 742
MTS Support 744
OS Authentication 745
Support for Oracle RAC 745
Support of Materialized Views 746
Stored Procedure Results 746
Configuring Connection Failover 748
Persisting a Result Set as an XML Data File 748
Isolation and Lock Levels Supported 748
ODBC Conformance Level 749
Number of Connections and Statements Supported 749
Using Arrays of Parameters 750

14 The SQL Server Classic Wire Protocol Driver 751

Driver Requirements 751
Windows 752
UNIX and Linux 752
Configuring and Connecting to Data Sources 752
Data Source Configuration in the UNIX odbc.ini File 753
Data Source Configuration through a GUI 754
Using a Connection String 759
Using a Login Dialog Box 761

Connection Option Descriptions 763
Windows 763
UNIX and Linux 765
Performance Considerations 783
Data Types 784
Unicode Support 786
Configuring Connection Failover 787
Isolation and Lock Levels Supported 787
Using The Snapshot Isolation Level 788
ODBC Conformance Level 788
Number of Connections and Statements Supported 789
Using Arrays of Parameters 789

15 The Driver for the Teradata Database 791

Driver Requirements 791
Configuring and Connecting to Data Sources 792
Data Source Configuration in the UNIX odbc.ini File 792
Data Source Configuration through a GUI 793
Using a Connection String 802
Using a Logon Dialog Box 803
Connection Option Descriptions 805
Data Types 826
Unicode Support 828
Persisting a Result Set as an XML Data File 829
Isolation and Lock Levels Supported 829
ODBC Conformance Level 829
Number of Connections and Statements Supported 829

Part 3: The 32-Bit Drivers

16 The Btrieve (Pervasive.SQL) Driver 833

Driver Requirements 834

Managing Databases 835

Transactions 835

Configuring and Connecting to Data Sources 836
Using a Connection String 841

Connection Option Descriptions 842

Defining Table Structure 849

Data Types 852

Indexes 854

Column Names 854

Select Statement 855

Alter Table Statement 856

Create and Drop Index Statements 857
Create Index 857
Drop Index 858

Isolation and Lock Levels Supported 858

ODBC Conformance Level 858

Number of Connections and Statements Supported 859

17 The dBASE Driver 861

Driver Requirements 861

Configuring and Connecting to Data Sources 862
Data Source Configuration in the UNIX odbc.ini File 862
Data Source Configuration through a GUI 863
dBASE 864
FoxPro 3.0 DBC 868
Using a Connection String 872

Connection Option Descriptions 873
Defining Index Attributes on Windows 884
Defining Index Attributes on UNIX and Linux 886
Data Types. 887
Column Names 889
Select Statement. 889
Alter Table Statement 890
Create and Drop Index Statements. 891
Create Index 891
Drop Index 893
Pack Statement. 894
SQL Statements for FoxPro 3.0 Database Containers. 895
Locking 896
Levels of Database Locking 896
Limit on Number of Locks 897
How Transactions Affect Record Locks 897
Isolation and Lock Levels Supported. 897
ODBC Conformance Level 898
Number of Connections and Statements Supported 898

18 The Informix Driver 899

Driver Requirements 900
Windows 900
UNIX (AIX, HP-UX PA-RISC, and Solaris) 900
Configuring and Connecting to Data Sources 901
Data Source Configuration in the UNIX odbc.ini File. . . . 902
Data Source Configuration through a GUI 903
Using a Connection String 908
Using a Logon Dialog Box 910
Connection Option Descriptions 912

Performance Considerations 922
Data Types 923
MTS Support 925
Persisting a Result Set as an XML Data File 927
Isolation and Lock Levels Supported 927
ODBC Conformance Level 928
Number of Connections and Statements Supported. 928

19 The Paradox Driver 929

Driver Requirements 929
Multiuser Access to Database Files 930
Configuring and Connecting to Data Sources 931
Using a Connection String. 935
Connection Options Descriptions 937
Data Types 943
Select Statement 944
Alter Table Statement 945
Create Table Statement. 946
Password Protection 947
Encrypting a Paradox Database File 948
Accessing an Encrypted Paradox Database File 948
Decrypting a Paradox Database File. 949
Removing a Password from Paradox 949
Removing All Passwords from Paradox 949
Index Files. 949
Primary Index 950
Non-Primary Index. 950

Create and Drop Index Statements 951
Create Index Primary Statement 952
Create Index Statement 952
Drop Index Statement 953

Transactions 954

Isolation and Lock Levels Supported 955

ODBC Conformance Level 955

Number of Connections and Statements Supported 955

20 The Text Driver. 957

Driver Requirements 957

Formats for Text Files 958

Configuring Data Sources 959
Data Source Configuration in the UNIX odbc.ini File. . . . 959
Data Source Configuration through a GUI 960
Using a Connection String 965

Connection Option Descriptions 967

Defining Table Structure on Windows 980

Defining Table Structure on UNIX and Linux 984

Date Masks 986

Data Types. 988

Select Statement. 989

Alter Table Statement 989

ODBC Conformance Level 990

Number of Connections and Statements Supported 990

21 The XML Driver 991

Driver Requirements 992

Supported Tabular Formats for XML Documents 992

Hierarchical-Formatted XML Document Support 993

Defining Locations. 996

Specifying Table Names in SQL Statements 998

Configuring and Connecting to Data Sources 1000
Using a Connection String. 1008
Using a Logon Dialog Box 1010

Connection Option Descriptions. 1010
Driver Setup Dialog Box Descriptions. 1011
Configure Location Dialog Box Descriptions 1019

Using Hints for Tabular-Formatted XML Documents 1028

Data Types 1031

Unicode Support 1035

Persisting a Result Set as an XML Data File 1035

ODBC Conformance Level 1035

Number of Connections and Statements Supported. 1036

SQL Supported 1036
SQL Statements 1036
Extensions to SQL Standards 1037
Grammar Token Definitions 1037

Index 1047

List of Tables

Table 2-1. Supported IP Address Formats 70
Table 2-2. Threading Information 73
Table 3-1. Summary: Failover and Related Connection Options 97
Table 3-2. Summary: Security Connection Options 108
Table 3-3. Pool Reset Connection Attributes 112
Table 3-4. Pool Statistics Connection Attribute 114
Table 3-5. Summary: Connection Pooling Connection Options 114
Table 3-6. Summary: Bulk Load Connection Options 130
Table 5-1. Kerberos Authentication Requirements for the DB2 Wire Protocol Driver 178
Table 5-2. DDF Work Classification Attributes 179
Table 5-3. DB2 Wire Protocol Attribute Names 209
Table 5-4. DB2 Data Types 253
Table 6-1. Greenplum Wire Protocol Attribute Names 277
Table 6-2. Greenplum Data Types 300
Table 7-1. Informix Wire Protocol Attribute Names 320
Table 7-2. Informix Data Types 332
Table 8-1. MySQL Wire Protocol Attribute Names 353
Table 8-2. MySQL Data Types 380
Table 9-1. Kerberos Authentication Requirements for the Oracle Wire Protocol Driver 386
Table 9-2. Oracle Wire Protocol Attribute Names 416
Table 9-3. Oracle Data Types 467

Table 10-1. PostgreSQL Wire Protocol Attribute Names 497
Table 10-2. PostgreSQL Data Types 526
Table 11-1. Kerberos Authentication Requirements for the SQL Server Native Wire Protocol Driver 536
Table 11-2. SQL Server Native Wire Protocol Attribute Names 562
Table 11-3. Microsoft SQL Server Data Types 600
Table 12-1. Kerberos Authentication Requirements for the Sybase Wire Protocol Driver 608
Table 12-2. Sybase Wire Protocol Attribute Names 639
Table 12-3. Sybase Data Types 684
Table 12-4. Summary of Fast and Slow Bulk Copy Mode Characteristics 692
Table 13-1. Oracle Attribute Names 713
Table 13-2. Oracle Data Types 737
Table 14-1. SQL Server Classic Wire Protocol Attribute Names on Windows . . 763
Table 14-2. SQL Server Classic Wire Protocol Attribute Names on UNIX/Linux . . 765
Table 14-3. Microsoft SQL Server Data Types 784
Table 15-1. Teradata Attribute Names 806
Table 15-2. Teradata Data Types 826
Table 15-3. Teradata Unicode Data Types 828
Table 16-1. Btrieve Attribute Names 843
Table 16-2. Btrieve Data Types 852
Table 17-1. dBASE Attribute Names 874
Table 17-2. dBASE Data Types 887
Table 17-3. Additional FoxPro 3.0 Data Types 888

Table 17-4. dBASE-Compatible Index Summary 893
Table 18-1. Informix Attribute Names 912
Table 18-2. Informix Data Types 923
Table 19-1. Paradox Attribute Names 937
Table 19-2. Paradox Data Types 943
Table 20-1. Common Text File Formats 958
Table 20-2. Text Attribute Names 967
Table 20-3. Date Masks for Text Driver 987
Table 20-4. Date Mask Examples 988
Table 20-5. Text Data Types 988
Table 21-1. Common Tabular Formats for XML Documents 992
Table 21-2. XML Attribute Names 1011
Table 21-3. XML Configure Location Attribute Names 1019
Table 21-4. Data Islands Data Types 1032
Table 21-5. ADO 2.5 Persisted Files Data Types 1033
Table 21-6. DataDirect Format Data Types 1034
Table 21-7. SQL Extensions 1037
Table 21-8. Reserved Keywords 1042

Preface

This book is your user's guide to Progress® DataDirect Connect® Series *for* ODBC from DataDirect Technologies, which includes the following products:

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

These products consist of a number of database *drivers* that are compliant with the Open Database Connectivity (ODBC) specification. A number of these drivers are available in both 32- and 64-bit formats. See the "Product Matrix" on page 27 for details.

Product Platforms

DataDirect Connect Series *for* ODBC drivers enable you to connect to a variety of databases from these platforms:

Windows (32-bit)

- Windows Server 2008
- Windows Vista
- Windows XP
- Windows Server 2003
- Windows 2000

Windows (64-bit)

- Windows Server 2008

- Windows Vista
- Windows XP Professional
- Windows Server 2003

UNIX and Linux (32-bit)

- AIX
- HP-UX aCC Enabled
- Linux
- Sun Solaris

UNIX and Linux (64-bit)

- AIX
- HP-UX aCC Enabled
- Linux
- Sun Solaris

See "Environment-Specific Information" on page 59 for detailed information regarding these platforms.

Product Matrix

The DataDirect Connect Series *for* ODBC products include 32- and 64-bit drivers. DataDirect Connect *for* ODBC (32-bit) and DataDirect Connect64 *for* ODBC (64-bit) are detailed in the following table. The Connect XE product consists of the 32-bit Driver for the Teradata Database and Connect64 XE of the 64-bit Driver for the Teradata Database.

Driver	Connect for ODBC	Connect64 for ODBC
DB2 Wire Protocol	X	X
Greenplum Wire Protocol	X	X
Informix Wire Protocol	X	X
MySQL Wire Protocol	X	X
Oracle Wire Protocol	X	X
PostgreSQL Wire Protocol	X	X
SQL Server Classic Wire Protocol	X	X
SQL Server Native Wire Protocol	X	X
Sybase Wire Protocol	X	X
Oracle (client)	X	X
Btrieve	X	
dBASE	X	
Informix (client)	X	
Paradox	X	
Text	X	
XML	X	

What's New in This Release?

The highlights of Release 6.0 Service Pack 2 are:

- New database version support
  - DB2 V9.7 for Linux, UNIX, Windows (LUW)
  - Greenplum 3.2 and 3.3
- New operating system support: Windows 7
- New SQL Server Native Wire Protocol driver
  - Wire Protocol driver
  - Unicode driver
  - DataDirect Application Failover
  - DataDirect Bulk Load
  - Data Encryption
  - Connection Pooling
  - Client Load Balancing
  - Query and Login Timeout
  - Microsoft SQL Server 2008 features:
    - New data type support
    - FILESTREAM attribute support
    - Sparse Column support
    - Transparent Data Encryption support
    - Table Compression support
    - New scalar function and SQL construct support

Using this Book

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

- Chapter 1 "Quick Start Connect" on page 41 explains the basics for quickly configuring and testing the drivers.
- Chapter 2 "Using The Product" on page 55 explains the drivers and ODBC, and discusses environment-specific subjects.
- Chapter 3 "Advanced Features" on page 85 explains at a general level advanced driver features such as failover, security, connection pooling, and bulk load.
- Chapter 4 "Configuring the Product on UNIX/Linux" on page 131 discusses UNIX and Linux environment variables and configuration of the drivers. It also provides a sample system information file, as well as discussing other driver tools for UNIX and Linux.
- A chapter for each database driver. Each driver's chapter is structured in the same way. First, it lists which versions of the databases the driver supports, the operating environments in which the driver runs, and the driver requirements for your operating environment. Next, it explains how to configure a data source and how to connect to that data source. Finally, the chapter provides information about data types, ODBC conformance levels, isolation and lock levels supported, and other driver-specific information.

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.

Database drivers are continually being added to each operating environment. For the latest information about the specific drivers available for your platform, refer to the DataDirect Technologies database support matrix Web pages at:

http://www.datadirect.com/products/odbc/matrix/connectodbc.htm (32-bit)

http://www.datadirect.com/products/odbc64/matrix/connect64odbc.htm (64-bit)

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

Conventions Used in This Book

The following sections describe the typography and other conventions used in this book.

Typographical Conventions

This book uses the following typographical conventions:

Convention	Explanation
italics	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**.

Convention	Explanation
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key. Also used for SQL reserved words.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values that you specify. For example, *`filename`*.
forward slash /	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.
vertical rule |	Indicates an "OR" separator used to delineate items.
brackets [ ]	Indicates optional items. For example, in the following statement: SELECT [DISTINCT], DISTINCT is an optional keyword. Also indicates sections of the Windows Registry.
braces { }	Indicates that you must select one item. For example, {yes | no} means that you must specify either yes or no.
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.

Environment-Specific Information

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

About the Product Documentation

The product library consists of the following books:

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

HTML Version

This library, except for the installation guide, is placed on your system as HTML-based online help during a normal installation of the product. It is located in the help subdirectory of the product

installation directory. To use online help, you must have one of the following Internet browsers installed.

- Internet Explorer 5.x, 6.x, and 7.x
- Mozilla Firefox 1.x, 2.x, and 3.x
- Netscape 4x, 7.x and 8.x
- Safari 1.x, 2.x, and 3x
- Opera 7.54u2, 8.x, and 9x

On Windows, you can access the entire Help system by selecting the help icon that appears in the DataDirect program group.

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

install_dir/help/help.htm

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/help.htm

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.

PDF Version

DataDirect product documentation is also provided in PDF format, which allows you to view it, perform text searches, or print it. You can view the PDF documentation using Adobe Reader. The PDF documentation is available on the product DVD and also on the DataDirect Technologies Web site:

http://www.datadirect.com/techres/odbcproddoc/index.ssp

You can download the entire library as a compressed file. When you uncompress the file, it appears in the correct directory structure.

If you want to copy the documentation library from the product DVD, you must maintain the same directory structure that is on the DVD.

- **To copy all product books**, copy the entire \books directory to your local or network drive.
- **To copy a specific set of books**, copy that book set's directory structure (beneath the \books directory) to your local or network drive. For example, in the case of:

  `\books\odbc`

  you would copy the entire `\odbc` directory.

NOTE: Maintaining the correct directory structure allows cross-book text searches and cross-references. If you download or copy the books individually outside of their normal directory structure, their cross-book search indexes and hyperlinked cross-references to other books will not work. You can view a book individually, but it will not open other books to which it has cross-references.

To help you navigate through the library, a file named **books.pdf** is provided. This file lists each online book provided for the product. We recommend that you open this file first and, from this file, open the book you want to view.

Contacting Technical Support

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

http://support.datadirect.com

The DataDirect Technologies 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.

To obtain technical support for an evaluation copy of the product, go to:

http://www.datadirect.com/support/eval_help/index.ssp

or contact your sales representative.

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

- 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 customer information, including location.
- The DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your DataDirect 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 recreated.

- 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.

Part 1: Getting Started

This part contains the following chapters:

- Chapter 1 “Quick Start Connect” on page 41
- Chapter 2 “Using The Product” on page 55
- Chapter 3 “Advanced Features” on page 85
- Chapter 4 “Configuring the Product on UNIX/Linux” on page 131

1 Quick Start Connect

This chapter provides basic information about configuring and test connecting with your drivers immediately after installation. To take full advantage of the features of the drivers, we recommend that you read "About the Product" on page 58 and the chapters specific to the drivers you are using.

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

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

This chapter contains the following sections:

- "Configuring and Connecting on Windows"
- "Configuring and Connecting on UNIX and Linux"
- "Using the Performance Wizard"

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 Through the Administrator

To configure a data source:

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

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

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

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

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

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

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

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

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

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

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

Testing the Connection

1 After you have configured the data source, you can click **Test Connect** on the Setup dialog box to attempt to connect to the data source using the connection options specified in the dialog box. Some drivers immediately return a message indicating success or failure. For most drivers, a logon dialog box appears as described in each individual driver chapter.

2 Supply the requested information in the logon dialog box and click **OK**. Note that the information you enter in the logon dialog box during a test connect is not saved.

   - If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

3 On the driver Setup dialog box, click **OK**. The values you have specified are saved and are the defaults used when you connect to the data source. You can change these defaults by using the previously described procedure to modify your data source. You can override these defaults by connecting to the data source using a connection string with alternate values. See individual driver chapters for information about using connection strings.

Configuring and Connecting on UNIX and Linux

The following basic information enables you to configure a data source and test connect with a driver immediately after installation. See Chapter 4 “Configuring the Product on UNIX/Linux” for detailed information about configuring the UNIX/Linux environment and data sources.

NOTE: The following data source configuration information assumes that you have a Motif Graphical User Interface (GUI) in your UNIX/Linux environment. If you do not, see “Configuration Through the odbc.ini File” on page 141 and follow the instructions for configuring the system information file (odbc.ini).

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

Environment Configuration

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, Borne, and equivalent shells, execute odbc.sh. For a C shell, execute odbc.csh. After running the setup script, execute:

   ```
   env
   ```

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

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

   From the C shell, you would enter:

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

Test Loading the Driver

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

For example, if the drivers are installed in /opt/odbc/lib, the following command attempts to load the 32-bit Oracle Wire Protocol 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 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 Through the Administrator

The DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) is located in the /tools directory of the product installation directory. For example:

```
/opt/odbc/tools/odbcadmin
```

To configure a data source:

1 To start the UNIX ODBC Administrator, change to the *install_dir*/tools directory, where *install_dir* is the path to the product installation directory. At a command prompt, enter:

- UNIX: `odbcadmin`
- Linux: ./`odbcadmin`

2 Click either the **User DSN** or **File DSN** tab to display a list of data sources.

- **User DSN**: 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.

- **File DSN**: 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.

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.

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

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

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

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

Testing the Connection

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

Using the Performance Wizard

The Performance Wizard leads you step-by-step through a series of questions about your application. Based on your answers, the Wizard provides the optimal settings for performance-related connection string options. The Wizard applies to the following drivers:

- DB2 Wire Protocol
- Informix Wire Protocol
- Oracle Wire Protocol
- PostgresSQL Wire Protocol
- Sybase Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle
- SQL Server Native Wire Protocol

The Wizard runs as an applet within a browser window. The browser must be configured to run applets. Refer to your browser's documentation for instructions on configuring your browser.

NOTE: Security features set in your browser can prevent the Performance Wizard from launching. If this is the case, a security warning message is displayed. Often, the warning message provides instructions for unblocking the Performance Wizard for the current session. To allow the Performance Wizard 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.

Starting the Wizard

You can start the Wizard in the following ways:

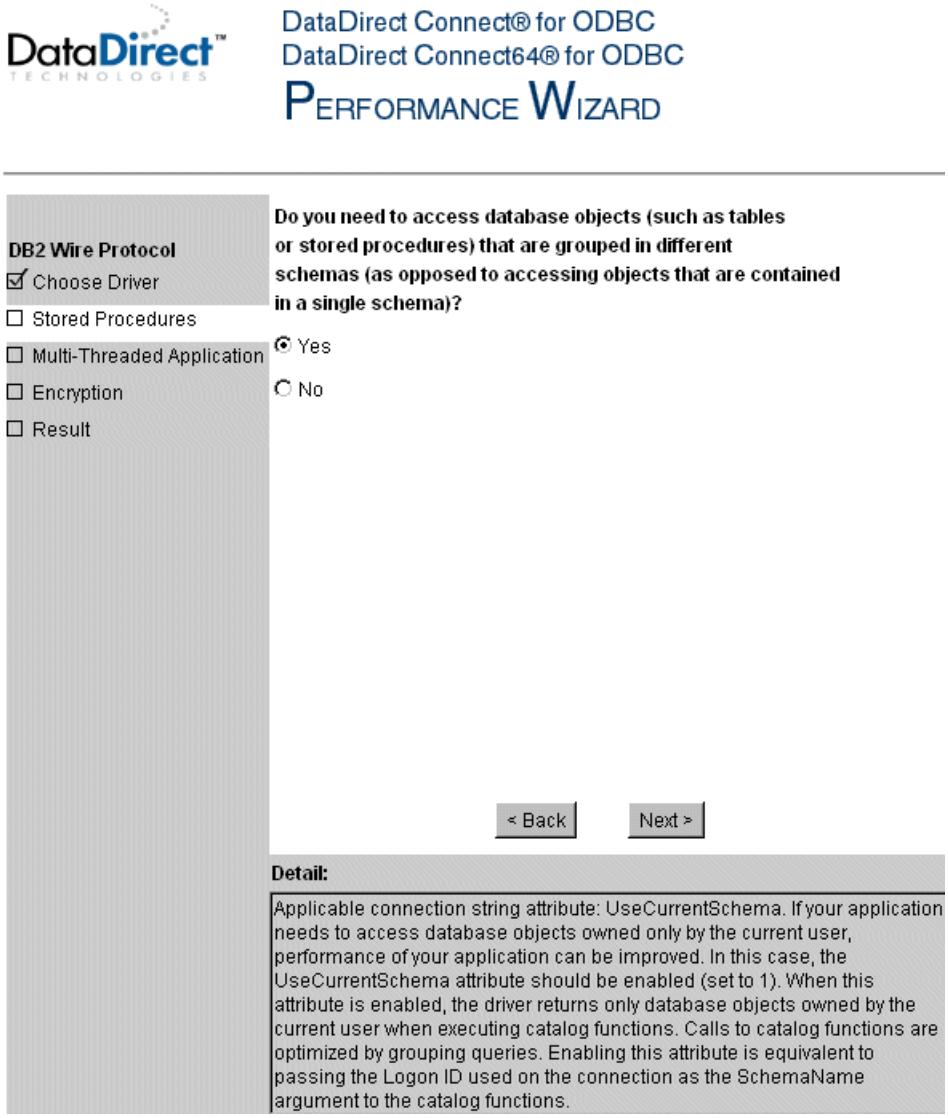
- On Windows, you can start the Wizard by selecting it from the product program group.
- On all platforms, you can start the Wizard by launching the following file from your browser window, where *install_dir* is your product installation directory:

  install_dir/wizards/index.html

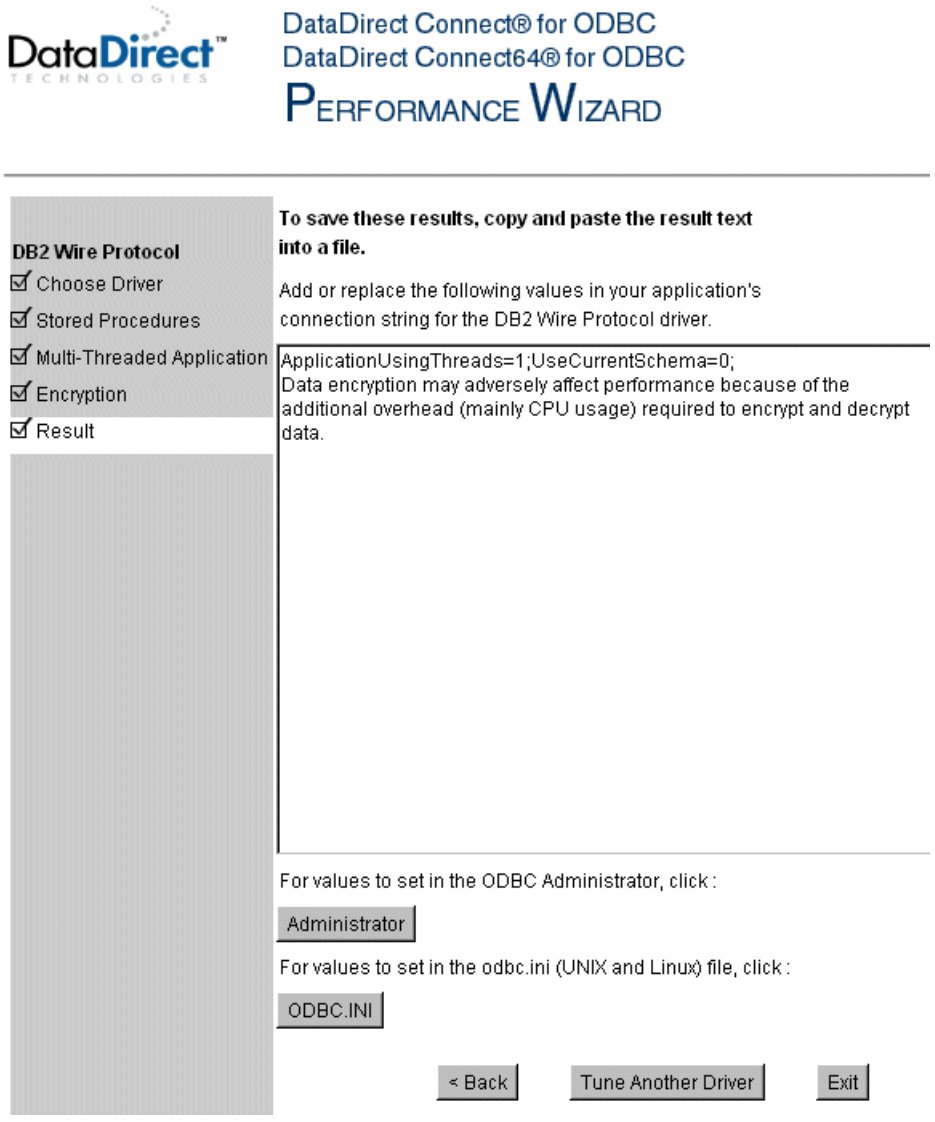
Tuning Performance Using the Wizard

After you start the Wizard, a Welcome window appears. Click **Start** to start the process and select a driver.

The following is an example of one of the questions you may be asked to answer for the DB2 Wire Protocol driver.



When you have answered all questions for a driver, the results appear in the form of a connection string, as shown in the following example:



You can copy these results to a connection string for immediate use or to a text file for later reference.

You can also either click **Administrator**, if you are using the Windows or UNIX ODBC administrator, or **ODBC.INI**, if you are editing the odbc.ini file. Clicking either of these buttons displays a window that provides the values to use for configuring a data source.

See “Data Source Configuration” on page 137 for details about configuring data sources through the odbc.ini file.

2 Using The Product

This chapter contains the following sections:

- "What Is ODBC?"
- "About the Product"
- "Environment-Specific Information"
- "Using IP Addresses"
- "Binding Parameter Markers"
- "Driver Threading Information"
- "Version String Information"
- "Retrieving Data Type Information"
- "Persisting a Result Set as an XML Data File"
- "Translators"

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:
  - Core functions that are based on the X/Open and SQL Access Group Call Level Interface specification
  - Extended functions that support additional functionality, including scrollable cursors
- 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. 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.

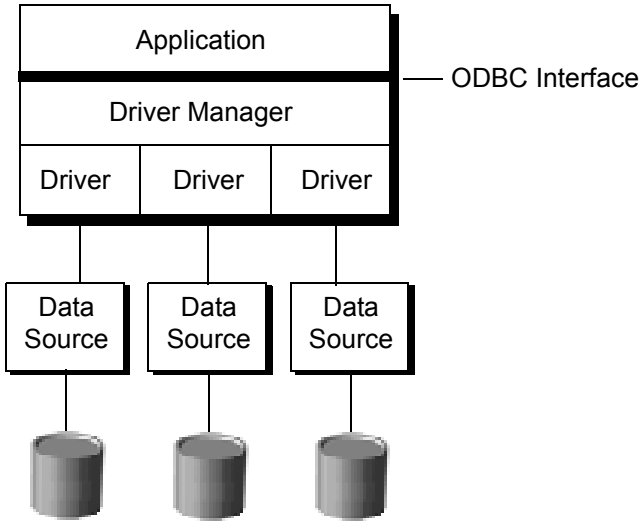
How Does It Work?

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 Product

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

DataDirect provides ODBC drivers for both relational and flat-file database systems. The flat-file drivers provide full SQL support; refer to Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference* for details.

Support for Multiple Environments

DataDirect Technologies provides ODBC-compliant database drivers for Windows, UNIX, and Linux operating systems. See "Environment-Specific Information" on page 59 for an explanation of the environment-specific differences when using the database drivers in your operating environment.

Database drivers are continually being added to each operating environment. For the latest information about the specific drivers available for your platform, refer to the DataDirect Technologies database support matrix Web pages at:

http://www.datadirect.com/products/odbc/matrix/connectodbc.htm (32-bit)

http://www.datadirect.com/products/odbc64/matrix/connect64odbc.htm (64-bit)

Environment-Specific Information

The sections "For Windows Users" on page 59 and "For UNIX and Linux Users" on page 62 contain information specific to your operating environment.

The following sections refer to threading models. Refer to Chapter 3 "Threading" in the *DataDirect Connect Series for ODBC Reference* for an explanation of threading.

For Windows Users

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

32-Bit Drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit drivers. If your application was built with 64-bit system libraries, you must use 64-bit drivers (see "64-Bit Drivers" on page 60). 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 DataDirect Connect *for* ODBC. All editions are supported unless otherwise noted.
  - Windows 7
  - Windows Server 2008

- Windows Vista
- Windows XP, Service Pack 1 and higher
- Windows Server 2003
- Windows 2000, Service Pack 1 and higher

■ The following operating systems are supported for DataDirect Connect XE *for* ODBC. All editions are supported unless otherwise noted.

- Windows Vista
- Windows XP Professional, Service Pack 2 and higher
- Windows Server 2003, Service Pack 2 and higher
- Windows 2000 Professional and Server, Service Pack 4

■ An application that is compatible with components that were built using Microsoft Visual Studio 2005 compiler version 8 and the standard Win32 threading model

■ You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

64-Bit Drivers

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

■ The following processors are supported:

- Intel Itanium II
- x64: Intel and AMD

■ The following operating systems are supported for DataDirect Connect64 *for* ODBC. All editions are supported unless otherwise noted.

- For Itanium II: Windows Server 2003

- For x64: Windows 7
- For x64: Windows Server 2008
- For x64: Windows Vista
- For x64: Windows Server 2003

  - For x64: Microsoft Windows XP Professional Edition

- The following operating systems are supported for DataDirect Connect64 XE *for* ODBC. All editions are supported unless otherwise noted.

  - For Itanium II: Windows Server 2003
  - For x64: Windows Vista
  - For x64: Windows Server 2003, Service Pack 2 and higher
  - For x64: Windows XP Professional, Service Pack 2 and higher

- For Itanium II: an application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 13.10.2240.8 and the standard Windows 64 threading model
- For x64: an application that is compatible with components that were built using Microsoft C/C++ Optimizing Compiler Version 14.00.40310.41 and the standard Windows 64 threading model
- You must have ODBC header files to compile your application. For example, Microsoft Visual Studio includes these files.

Setup of the Drivers

The drivers must be configured before they can be used. See Chapter 1 “Quick Start Connect” on page 41 for information about using the Windows ODBC Administrator. See the individual driver chapters for details about driver configuration.

Driver Names

The prefix for all 32-bit driver file names is IV. The prefix for all 64-bit driver file names is DD. The file extension is .DLL, which indicates dynamic link libraries. For example, the 32-bit DB2 Wire

Protocol driver file name is IVDB2*nn*.DLL, where *nn* is the revision number of the driver.

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

For UNIX and Linux Users

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

32-Bit Drivers

- All required network software that is supplied by your database system vendors must be 32-bit compliant.
- If your application was built with 32-bit system libraries, you must use 32-bit drivers. If your application was built with 64-bit system libraries, you must use 64-bit drivers (see "64-Bit Drivers" on page 65). The database to which you are connecting can be either 32-bit or 64-bit enabled.

AIX

- IBM POWER processor
- AIX 5L operating system, versions 6.1, 5.3, 5.2, and 5.1
- An application compatible with components that were built using CSet++ 5.0 and the AIX native threading model

NOTE FOR TERADATA USERS: When compiling an application on AIX for use with the driver for the Teradata database, you must use the `-brtl` option. For example:

```
cc -o pgm pgm.o  -brtl -lodbc
```

or

```
ld -o pgm -brtl pgm.o -lodbc
```

HP-UX

- The following processors are supported:
  - PA-RISC
  - Intel Itanium II (IPF)
- The following operating systems are supported:
  - For PA-RISC: HP-UX 11i Versions 3 and 2 (B.11.31 and B.11.23), 11i (B.11.11), and 11
  - For IPF: HP-UX IPF 11i Versions 3 and 2 (B.11.31 and B.11.23)
- For PA-RISC: An application compatible with components that were built using HP aC++ 3.30 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads)
- For IPF: An application compatible with components that were built using HP aC++ 5.36 and the HP-UX 11 native (kernel) threading model (posix draft 10 threads)

NOTE: DataDirect Connect XE *for* ODBC is not supported on IPF. Only the following DataDirect Connect *for* ODBC drivers are supported on IPF:

- DB2 Wire Protocol
- Informix Wire Protocol
- MySQL Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Classic Wire Protocol
- SQL Server Native Wire Protocol
- Sybase Wire Protocol
- Oracle

Linux

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

- The following operating systems are supported:
  - Red Hat Enterprise Linux 5.0 and 4.0
  - SUSE Linux Enterprise Server 10.0
- 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).

NOTE: All drivers are supported on Linux except for the Informix driver.

Solaris

- The following processors are supported:
  - Sun SPARC
  - x86: Intel
  - x64: Intel and AMD
- The following operating systems are supported:
  - For Sun SPARC: Sun Solaris 10, 9, and 8
  - For x86/x64: Sun Solaris 10
- For Sun SPARC: An application compatible with components that were built using Sun Workshop v. 6 update 2 and the Solaris native (kernel) threading model
- For x86/x64: An application compatible with components that were built using Sun C++ 5.8 and the Solaris native (kernel) threading model

NOTE: DataDirect Connect XE *for* ODBC is not supported on x86/x64. Only the following DataDirect Connect *for* ODBC drivers are supported on x86/x64:

- DB2 Wire Protocol
- MySQL Wire Protocol
- SQL Server Classic Wire Protocol
- SQL Server Native Wire Protocol

- Oracle Wire Protocol
- Sybase Wire Protocol
- PostgreSQL Wire Protocol

64-Bit Drivers

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, versions 6.1, 5.3, 5.2, and 5.1
- An application compatible with components that were built using Visual Age C++ version 5.0.2.0 and the AIX native threading model

HP-UX

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

Linux

- The following processors are supported:
  - Intel Itanium II
  - x64: Intel and AMD
- The following operating systems are supported:

- For Itanium II: Red Hat Enterprise Linux AS, ES, and WS versions 5.0 and 4.0
- For x64: Red Hat Enterprise Linux AS, ES, and WS version 5.0 and 4.0

  NOTE: The Oracle (client) driver is not supported on the Red Hat x64 operating system.
- For x64: SUSE Linux Enterprise Server 10.0

- For Itanium II: an application compatible with components that were built using g++ GNU project C++ Compiler version 3.3.2 and the Linux native pthread threading model (Linuxthreads)
- For x64: an application compatible with components that were built using g++ GNU project C++ Compiler version 3.4 and the Linux native pthread threading model (Linuxthreads)

NOTE: DataDirect Connect64 XE *for* ODBC is not supported on Linux Itanium II.

Solaris

- The following processors are supported:
  - Sun SPARC
  - x64: Intel and AMD
- The following operating systems are supported:
  - For Sun SPARC: Solaris 10, 9, and 8 operating systems.
  - For x64: Solaris 10 operating system

- For Sun SPARC: An application compatible with components that were built using Sun Workshop v. 6 update 2 and the Solaris native (kernel) threading model
- For x64: An application compatible with components that were built using Sun C++ Compiler version 5.8 and the Solaris native (kernel) threading model

NOTE: DataDirect Connect64 XE *for* ODBC is not supported on Solaris x64. Only the following drivers are supported on Solaris x64:

- DB2 Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Classic Wire Protocol
- SQL Server Native Wire Protocol
- Sybase Wire Protocol

AIX

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and "The demoodbc Application" on page 160 for details.

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

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

HP-UX 11 aCC

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

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

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and "The demoodbc Application" on page 160 for details. You must also include the +DD64 compiler switch if you are building 64-bit binaries. For example, to build demoodbc, you would use:

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

Linux

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and "The demoodbc Application" on page 160 for details.

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

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

Solaris

If you are building 64-bit binaries, you must pass the define ODBC64. Demoodbc provides an example of this. See the installed file demoodbc.txt and "The demoodbc Application" on page 160 for details.

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

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

Setup of the Environment and the Drivers

On UNIX and Linux, several environment variables and the system information file must be configured before the drivers can be used. See Chapter 1 "Quick Start Connect" on page 41 for a brief description of these variables and information about using the DataDirect ODBC Data Source Administrator for UNIX/Linux. See the individual driver chapters for details about driver configuration. See Chapter 4 "Configuring the Product on UNIX/Linux" on page 131 for complete information about using the drivers on UNIX and Linux.

Driver Names

The drivers are ODBC API-compliant dynamic link libraries, referred to in UNIX and Linux as shared objects. The prefix for all 32-bit driver file names is iv. The prefix for all 64-bit driver file names is dd. The driver file names are lowercase and the extension is .so, the standard form for a shared object. For example, the 32-bit DB2 Wire Protocol driver file name is ivdb2*nn*.so, where *nn* is the revision number of the driver. For drivers on HP-UX PA-RISC only, the extension is .sl, for example, ivdb2*nn*.sl.

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

Using IP Addresses

The DataDirect Connect *for* ODBC Wire Protocol drivers support Internet Protocol (IP) addresses in IPv4 and IPv6 format. IPv6 addresses are only supported when connecting to certain database versions, as shown in Table 2-1.

Table 2-1. Supported IP Address Formats

Driver	IPv4	IPv6
DB2 Wire Protocol	All supported versions	DB2 9.1 for Linux/UNIX/Windows and higher DB2 V5R2 for iSeries and higher
Greenplum Wire Protocol	All supported versions	not supported
Informix Wire Protocol	All supported versions	Informix 10 and higher
MySQL	All supported versions	not supported
Oracle Wire Protocol	All supported versions	not supported

Table 2-1. Supported IP Address Formats *(cont.)*

Driver	IPv4	IPv6
Microsoft SQL Server Classic Wire Protocol (UNIX/Linux only)	All supported versions	Microsoft SQL Server 2005 and higher
Microsoft SQL Server Native Wire Protocol	All supported versions	Microsoft SQL Server 2005 and higher
PostgreSQL Wire Protocol	All supported versions	PostgreSQL 8.2 and higher
Sybase Wire Protocol	All supported versions	Sybase 12.5.2 and higher

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 DB2 server is specified in IPv6 format:

```
DRIVER=DataDirect DB2 Wire Protocol;
IpAddress=2001:DB8:0000:0000:8:800:200C:417A;PORT=5179;
DB=DB2ACCT;UID=JOHN;PWD=XYZZYYou
```

In addition to the normal IPv6 format, the DataDirect Connect *for* ODBC drivers 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 DB2 Wire Protocol;
IpAddress=2001:DB8:0:0:8:800:200C:417A;PORT=5179;
DB=DB2ACCT;UID=JOHN;PWD=XYZZYYou
```

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

```
DRIVER=DataDirect DB2 Wire Protocol;
IpAddress=2001:DB8:0:0:8:800:123.456.78.90;PORT=5179;
DB=DB2ACCT;UID=JOHN;PWD=XYZZYYou
```

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

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

Binding Parameter Markers

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

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

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

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

To ensure interoperability, the DataDirect Connect Series *for* ODBC drivers use only the parameter binding information that is provided by the application. Some DBMSs cannot publish dynamic parameter information back to an ODBC driver. For example, both the Microsoft SQL Server and Oracle databases can determine that a parameter is an integer; however, the Oracle query processor cannot publish this information back to the driver.

Driver Threading Information

Table 2-2 summarizes the threading information available at this time for the drivers. Always consult the readme file for the most up-to-date information as threading information is subject to change with new database transport and server revisions. Currently, the XML driver is the only thread-impaired driver. Refer to Chapter 3 "Threading" in the *DataDirect Connect Series for ODBC Reference* for more information about threading and how to obtain treading information through SQLGetInfo.

Table 2-2. Threading Information

Driver	Fully Threaded	Thread Per Connect
Btrieve	X	
dBASE	X	
DB2 Wire Protocol		X
Greenplum Wire Protocol		X
Informix Wire Protocol		X
Informix		X
MySQL Wire Protocol		X
Oracle Wire Protocol		X
Oracle		X

Table 2-2. Threading Information *(cont.)*

Driver	Fully Threaded	Thread Per Connect
Paradox	X	
PostgreSQL Wire Protocol		X
SQL Server Classic Wire Protocol		X
SQL Server Native Wire Protocol		X
Sybase Wire Protocol		X
Teradata	X	
Text	X	

Version String Information

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

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

or

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

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

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

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

XX.YY.ZZZZ(U*BBBB*)

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

XX.YY.ZZZZ

where:

XX is the major version of the product.

YY is the minor version of the product.

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

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

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

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

For example:

```
06.00.0002 (b0001, u0002, F0001)
      |__|  |___|  |___|  |___|
     Driver  Bas    Utl    Flt
```

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

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

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

The syntax for the tool is:

```
ivtestlib shared_object
```

or

```
ddtestlib shared_object
```

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

```
ivtestlib ivora24.so
```

returns:

```
06.00.0001 (B0002, U0001)
```

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

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

```
ddtestlib libodbc.so
```

returns:

```
06.00.0001 (U0001)
```

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

```
ivtestlib libivicu24.so
```

```
06.00.0001
```

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

getFileVersionString Function

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

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

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

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

Retrieving Data Type Information

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

On Windows, you can use ODBC Test to call SQLGetTypeInfo against the ODBC data source to return the data type information. Refer to Chapter 1 "Diagnostic Tools" in the *DataDirect Connect Series for ODBC Troubleshooting Guide* for details about ODBC Test.

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

```
void ODBC_GetTypeInfo(SQLHANDLE hstmt, SQLSMALLINT dataType)
{

    RETCODE rc;

// There are 19 columns returned by SQLGetTypeInfo.
// This example displays the first 3.
// Check the ODBC 3.x specification for more information.
```

```
// Variables to hold the data from each column
    char             typeName[30];
    short            sqlDataType;
    unsigned long    columnSize;

    SQLINTEGER       strlenTypeName,
                     strlenSqlDataType,
                     strlenColumnSize;

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

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

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

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

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

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

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

Persisting a Result Set as an XML Data File

The DataDirect Connect Series *for* ODBC drivers allow you to persist a result set as an XML data file with embedded schema. To implement XML persistence, a client application must do the following:

1 Turn on STATIC cursors. For example:

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

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

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

2 Execute a SQL statement. For example:

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

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

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

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

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

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

```
Function Sequence Error
```

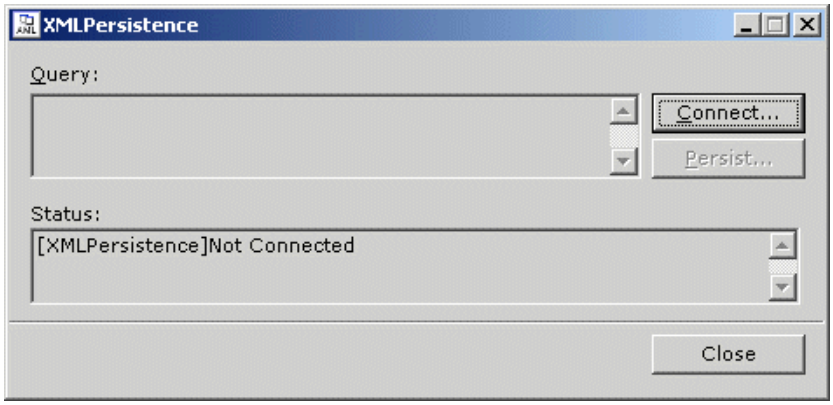
Using the Windows XML Persistence Demo Tool

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

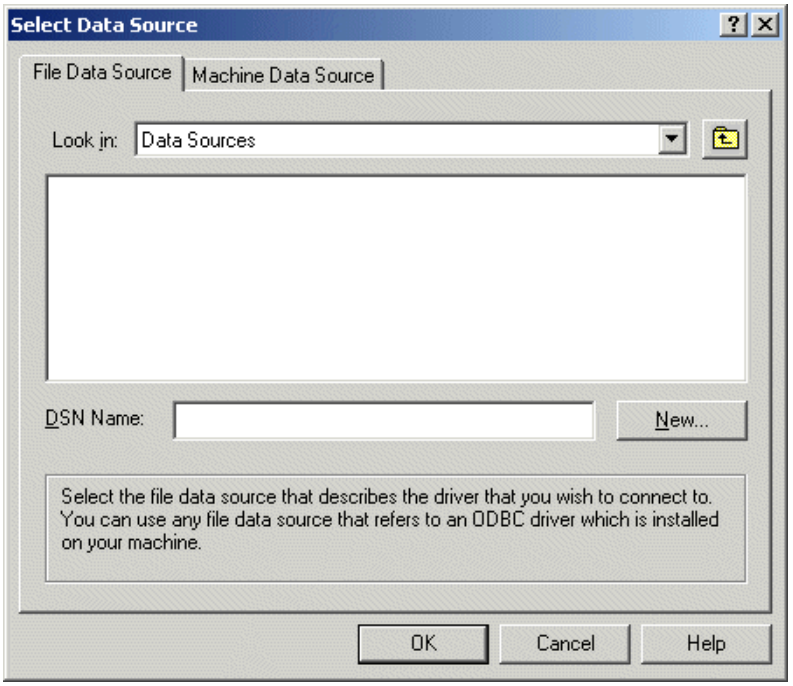
The tool has a graphical user interface and allows you to persist data as an XML data file.

To use the Windows XML Persistence Demo tool:

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



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



3 You must either select an existing data source or create a new one. Take one of the following actions:

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

4 After you have connected to a database, type a SQL Select statement in the Query text box of the XML Persistence dialog box. Then, click **Persist**. The Save As dialog box appears.

5 Specify a name and location for the XML data file that will be created. Then, click **OK**.

Note that the Status box in the XML Persistence dialog box displays whether the action failed or succeeded.

6 Click **Disconnect** to disconnect from the database.

7 Click **Close** to exit the tool.

Using the UNIX/Linux XML Persistence Demo Tool

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

Translators

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

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

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

3 Advanced Features

This chapter contains the following sections:

- "Using Failover"
- "Using Security"
- "Using DataDirect Connection Pooling"
- "Using DataDirect Bulk Load"

Using Failover

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

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

it provides protection for Select statements that have work in progress. If a connection to the database is lost, the driver fails over the connection to an alternate server, preserving the state of the connection at the time it was lost and preserving the state of any work being performed by Select statements.

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

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

- "Connection Failover" on page 87
- "Extended Connection Failover" on page 89
- "Select Connection Failover" on page 92

Regardless of the failover method you choose, you must configure one or multiple alternate servers using the Alternate Servers connection option. See "Guidelines for Primary and Alternate Servers" on page 94 for information about primary and alternate servers.

Connection Failover

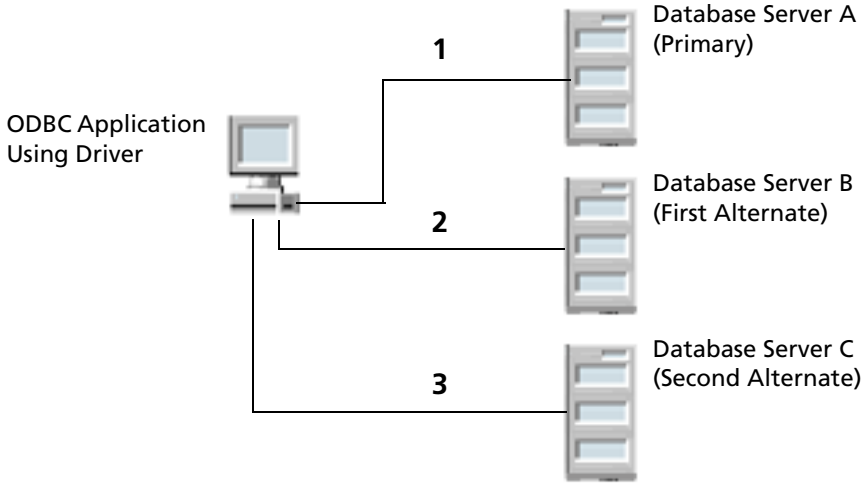
Connection failover is available in the following the DataDirect Connect Series *for* ODBC drivers:

- DB2 Wire Protocol
- Informix Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Native Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle
- Sybase Wire Protocol
- SQL Server Classic Wire Protocol (UNIX only)

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

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

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



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

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the driver can retry each alternate database server (primary and alternate) for a specified number of attempts. You can specify the number of attempts that are made through the *connection retry* feature. You can also specify the number of seconds of delay, if any, between attempts through the *connection delay* feature. See "Using Connection Retry" on page 96 for more information about connection retry.

A driver fails over to the next alternate database server only if a successful connection cannot be established with the current alternate server. If the driver successfully establishes communication with a database server and the connection request is rejected by the database server because, for example, the login information is invalid, then the driver generates an error and does not try to connect to the next database server in

the list. It is assumed that each alternate server is a mirror of the primary and that all authentication parameters and other related information are the same.

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

Extended Connection Failover

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

- DB2 Wire Protocol
- MySQL Wire Protocol
- PostgreSQL Wire Protocol
- Sybase Wire Protocol
- Greenplum Wire Protocol
- Oracle Wire Protocol
- SQL Server Native Wire Protocol

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

- New connections, in the same way as described in "Connection Failover" on page 87
- 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 properties 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
- Restores manual commit mode if the connection was in manual commit mode at the time of the failover.

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

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

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

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

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

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

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

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

Select Connection Failover

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

- DB2 Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Native Wire Protocol
- Sybase Wire Protocol

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

- New connections, in the same way as described in "Connection Failover" on page 87
- Lost connections, in the same way as described in "Extended Connection Failover" on page 89.

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.

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

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

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

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

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

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

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

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

Guidelines for Primary and Alternate Servers

Many databases provide advanced database replication technologies such as DB2 High Availability Disaster Recovery (HADR) and Oracle Real Application Clusters (RAC). The failover functionality provided by the drivers does not require any of these technologies, but can work with them to provide comprehensive failover protection. Use the following guidelines for primary and alternate servers to ensure that failover works correctly in your environment:

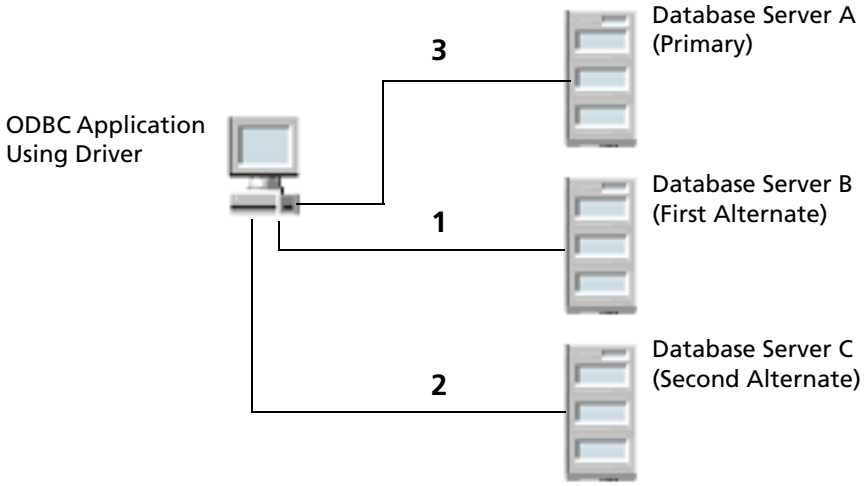
- Alternate servers should mirror data on the primary server or be part of a configuration where multiple database nodes share the same physical data.
- If using failover with DB2 HADR, the primary server must be the primary server configured in your HADR system and any alternate server must be a standby server configured in your HADR system.

Using Client Load Balancing

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

- DB2 Wire Protocol
- Informix Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Native Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle
- Sybase Wire Protocol
- SQL Server Classic Wire Protocol (UNIX only)

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, let us suppose that client load balancing is enabled as shown in the following illustration:



First, Database Server B is tried (**1**). Then, Database Server C may be tried (**2**), followed by a connection attempt to Database Server A (**3**). In contrast, if client load balancing were not

enabled in this scenario, each database server would be tried in sequential order, primary server first, then each alternate server based on its entry order in the alternate servers list.

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

Using Connection Retry

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

- DB2 Wire Protocol
- Informix Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Native Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle
- Sybase Wire Protocol
- SQL Server Classic Wire Protocol (UNIX only)

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

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

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

Summary of Failover-Related Options

Table 3-1 summarizes how failover-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options. Not all options are available in every failover-enabled driver. The step numbers in the table refer the procedure that follows the table

Table 3-1. Summary: Failover and Related Connection Options

Option	Characteristic
Alternate Servers (see Step 1)	One or multiple alternate database servers. An IP address or server name identifying each server is required.
Connection Retry Count (see Step 5)	Number of times the driver retries the primary database server, and if specified, alternate servers until a successful connection is established.
Connection Retry Delay (see Step 6)	Wait interval, in seconds, between connection retry attempts when the Connection Retry Count option is set to a positive integer.
Failover Granularity (see Step 3)	The type of behavior that the driver exhibits when errors are detected during the failover process.
Failover Mode (see Step 2)	The type of failover that the driver attempts.

Table 3-1. Summary: Failover and Related Connection Options (cont.)

Option	Characteristic
Failover Preconnect (see Step 4)	Determines whether the driver makes a connection attempt to the next server in the Alternate Servers list at the time of the initial connection.
Load Balancing (see Step 7)	Determines whether the driver uses client load balancing in its attempts to connect to primary and alternate database servers. In this case, the driver attempts to connect to the database servers in random order.

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

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

the failover process, but generates a warning that the Select statement must be reissued.

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

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

4. Optionally, enable the Failover Preconnect connection option (FailoverPreconnect=1) if you want the driver to establish a connection with the primary and an alternate server at the same time. This value applies only when Failover Mode is set to Extended Connection (FailoverMode=1) or Select (FailoverMode=2). The default behavior is to connect to an alternate server only when failover is caused by an unsuccessful connection attempt or a lost connection (FailoverPreconnect=0).

5. Optionally, specify the number of times the driver attempts to connect to the primary and alternate database servers after the initial unsuccessful connection attempt. By default, the driver does not retry. To set this feature, use the Connection Retry Count connection option.

6. Optionally, specify the wait interval, in seconds, between attempts to connect to the primary and alternate database servers. The default interval is 3 seconds. To set this feature, use the Connection Retry Delay connection option.

7. Optionally, specify whether the driver will use client load balancing in its attempts to connect to primary and alternate database servers. If load balancing is enabled, the driver uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load

balancing. To set this feature, use the Load Balancing connection option.

A Connection String Example

The following connection string configures the Oracle Wire Protocol 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 Oracle Wire Protocol driver to use connection failover in conjunction with some of its optional features in your odbc.ini file, you could set the following connection string attributes:

```
Driver=ODBCHOME/lib/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 Security

The drivers support authentication and data encryption.For current information, refer to the security matrix on the DataDirect Technologies Web site:

http://www.datadirect.com/products/security/documentation/securitymatrix.htm

The individual driver chapters provide driver-specific details, but the following sections give an overview of both authentication and data encryption, as well as discussing general requirements.

Authentication

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

- *User ID/password authentication* authenticates the user to the database using a database user name and password.
- *Client authentication* uses the user ID and password of the user logged onto the system on which the driver is running to authenticate the user to the database. The database server relies on the client to authenticate the user and does not provide additional authentication.
- *Kerberos authentication* is a trusted third-party authentication service that verifies user identities. DataDirect Connect Series *for* ODBC supports both Windows Active Directory Kerberos and MIT Kerberos implementations.

Kerberos and NTLM Authentication

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

- DB2 Wire Protocol
- SQL Server Native Wire Protocol
- Driver for the Teradata Database
- Oracle Wire Protocol
- Sybase Wire Protocol

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

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

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

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

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

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

where *user* is the application user.

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

On Windows platforms only, NTLM authentication is available with the SQL Server Native Wire Protocol driver and the driver for the Teradata database.

Data Encryption Across the Network

If your database connection is not configured to use data encryption, data is sent across the network in a format that is designed for fast transmission and can be decoded by

interceptors, given some time and effort. For example, text data is often sent across the wire as clear text. Because this format does not provide complete protection from interceptors, you may want to use data encryption to provide a more secure transmission of data. For example, you may want to use data encryption in the following scenarios:

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

Certain DataDirect Connect Series *for* ODBC drivers support Secure Sockets Layer (SSL). SSL is an industry-standard protocol for sending encrypted data over database connections. SSL secures the integrity of your data by encrypting information and providing client/server authentication. In addition, the DataDirect Connect Series *for* ODBC Wire Protocol driver supports DB2 database-specific encryption.

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

SSL Encryption

SSL Encryption is available in the following DataDirect Connect Series *for* ODBC drivers:

- DB2 Wire Protocol
- Oracle Wire Protocol
- SQL Server Native Wire Protocol
- MySQL Wire Protocol
- PostgreSQL Wire Protocol
- Sybase Wire Protocol

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*. The drivers support SSL2, SSL3 and TLS1, and negotiate the highest SSL/TLS protocol available during the handshake. The result of this negotiation determines the encryption cipher suite to be used for the SSL session.

The encryption cipher suite defines the type of encryption that is used for any data exchanged through an SSL connection. Refer to Chapter 8 "SSL Encryption Cipher Suites" in the *DataDirect Connect Series for ODBC Reference* for a list of the encryption cipher suites supported by the drivers.

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. Not all databases support SSL client authentication.

Certificates

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

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

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.

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

Summary of Security-Related Options

Table 3-2 summarizes how security-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 3-2. Summary: Security Connection Options

Option	Characteristic
Authentication Method	The method the driver uses to authenticate the user to the server when a connection is established.
Encryption Method	The method the driver uses to encrypt data sent between the driver and the database server.
GSS Client Library	The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).
Host Name In Certificate	The host name established by the SSL administrator for the driver to validate the host name contained in the certificate.
Key Password	The password required to access an individual key in the keystore.
Keystore	The path that specifies the location of the keystore file.
Keystore Password	The password required to access the keystore.
Truststore	The path that specifies the location of the truststore file.
Truststore Password	The password required to access the truststore.
User Name	The default user ID used to connect to your database.
Validate Server Certificate	Validates the security certificate of the server as part of the SSL authentication handshake.

Using DataDirect Connection Pooling

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

- DB2 Wire Protocol
- Greenplum Wire Protocol
- MySQL Wire Protocol
- Oracle Wire Protocol
- PostgreSQL Wire Protocol
- SQL Server Native Wire Protocol
- Sybase Wire Protocol

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

NOTE: Connection pooling works only with connections that are established using SQLDriverConnect with the SQL_DRIVER_NO_PROMPT argument.

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

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

Creating a Connection Pool

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

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

Adding Connections to a Pool

A connection pool is created in the process of creating each unique connection string that an application uses. When a pool is created, it is populated with enough connections to satisfy the minimum pool size requirement, set by the Min Pool Size connection option. The maximum pool size is set by the Max Pool Size connection option. 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 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 3-3. Pool Reset Connection Attributes

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

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

Handling Dead Connections in a Pool

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

DataDirect Connect Series *for* ODBC drivers handle this situation transparently to the user. The application does not receive any

errors on the attempt because the driver simply returns a connection from a connection pool. The first time the connection handle is used to execute a SQL statement, the driver detects that the physical connection to the server has been lost and attempts to reconnect to the server *before* executing the SQL statement. If the driver can reconnect to the server, the result of the SQL execution is returned to the application; no errors are returned to the application.

The driver uses connection failover option values, if they are enabled, when attempting this seamless reconnection; however, it attempts to reconnect even if these options are not enabled. See "Connection Failover" on page 87 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 DataDirect Technologies uses for handling dead connections in connection pools allows for maximum performance of the connection pooling mechanism. Some drivers periodically test the server with a dummy SQL statement while the connections sit idle. Other drivers test the server when the application requests the use of the connection from the connection pool. Both of these approaches add round trips to the database server and ultimately slow down the application during normal operation.

Connection Pool Statistics

DataDirect has created a connection attribute to monitor the status of the DataDirect Connect Series *for* ODBC connection pools. This attribute allows your application to fetch statistics for the pool to which a connection belongs.

Table 3-4. Pool Statistics Connection Attribute

Connection Attribute	Description
SQL_ATTR_POOL_INFO Value: SQL_GET_POOL_INFO	Calling SQLGetConnectAttr (SQL_ATTR_POOL_INF, SQL_GET_POOL_INFO) returns a poolStruct that contains the statistics for the connection pool to which this connection belongs. This is a write-only connection attribute. The driver returns an error if SQLGetConnectAttr (SQL_ATTR_POOL_INFO) is called.

Summary of Pooling-Related Options

Table 3-5 summarizes how connection pooling-related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 3-5. Summary: Connection Pooling Connection Options

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

Using DataDirect Bulk Load

DataDirect Bulk Load is available in the following DataDirect Connect Series *for* ODBC drivers:

- DB2 Wire Protocol
- Oracle Wire Protocol
- SQL Server Native Wire Protocol
- Sybase Wire Protocol

DataDirect Bulk Load is a feature that allows your application to send large numbers of rows of data to the database in a continuous stream instead of in numerous smaller database protocol packets. Although similar to parameter array batch operations, performance improves 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.

IMPORTANT: Because a bulk load operation bypasses 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 does not return an error; your application must provide its own data integrity checks.

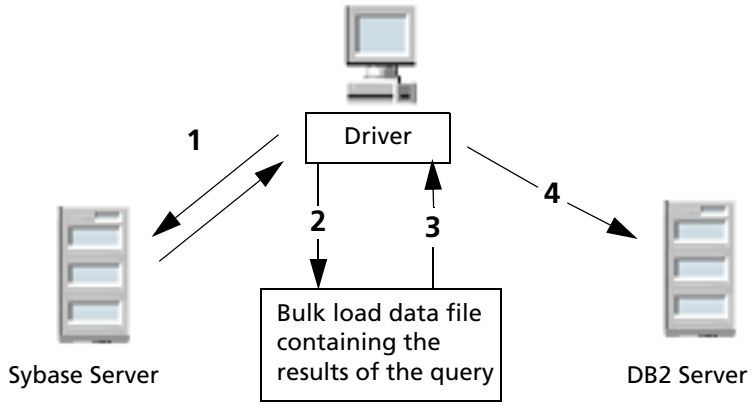
DataDirect Bulk Load provides a simple method to do bulk load operations across databases without altering your application; otherwise, you would have to deal with the different proprietary tools of each database vendor. DataDirect Bulk Load works in a consistent way for all DataDirect Connect products that support bulk load functionality.

NOTE: DataDirect Bulk Load requires a licensed installation of the drivers for full bulk load functionality. 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 DataDirect SupportLink for further information.

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

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



1. Application using Sybase Wire Protocol driver sends query to and receives results from Sybase server.
2. Driver exports results to bulk load data file.
3. Driver retrieves results from bulk load data file.
4. Driver bulk loads results on 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 the individual driver chapters for a description of the Bulk tab.

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

NOTE: 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 and in "DataDirect Bulk Load Functions" in Chapter 9 of the *DataDirect Connect Series for ODBC Reference*

Exporting Data from a Database

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

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

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

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

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

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

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

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

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

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

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

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

Refer to "DataDirect Bulk Load Functions" in Chapter 9 of the *DataDirect Connect Series for ODBC Reference* for a full description of these functions.

Your application can export a result set using the DataDirect statement attributes SQL_BULK_EXPORT and SQL_BULK_EXPORT_PARAMS. Refer to "DataDirect Bulk Load Statement Attributes" in Chapter 9 of the *DataDirect Connect Series for ODBC Reference* for a full description of these attributes.

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

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

A bulk load configuration file with and .xml extension is also created when either a table or a result set is exported to a bulk load data file. See "The Bulk Load Configuration File" on page 123 for an example of a bulk load configuration file.

In addition, a log file of events as well as external overflow files can be created during a bulk export operation. The log file is configured through either the driver Setup dialog Bulk tab, the ExportTableToFile function, or the SQL_BULK_EXPORT statement attribute. The external overflow files are configured through connection options; see "External Overflow Files" on page 129 for details.

Bulk Loading to a Database

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

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

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

```
HDBC      hdbc;
HENV      henv;
void      *driverHandle;
HMODULE   hmod;
PLoadTableFromFile loadTableFromFile;

char      tableName[128];
char      fileName[512];
char      configFile[512];
char      logFile[512];
char      discardFile[512];
int       errorTolerance;
int       warningTolerance;
int       loadStart;
int       loadCount;
int       readBufferSize;

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

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

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

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

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

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

Refer to "DataDirect Bulk Load Functions" in Chapter 9 of the *DataDirect Connect Series for ODBC Reference* for a full description of these functions.

NOTE FOR SYBASE USERS: Additional database configuration is required for destination tables that do not have an index. See "Using DataDirect Bulk Load on Sybase" on page 691 for more information.

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.

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 nullablity for all types.

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

- All data is read in as character data. Each value between commas is read as character data.

- The default character set is defined, on Windows, by the current Windows code page. On UNIX/Linux, it is the IANAAppCodePage value, which defaults to 4.

For example, the format of the bulk load data file GBMAXTABLE.csv (discussed in "Exporting Data from a Database" on page 118) 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://www.datadirect.com/ns/bulk/BulkData.xsd" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <column datatype="DECIMAL" precision="38" scale="0" nullable=
      "false">INTEGERCOL</column>
    <column datatype="VARBINARY" length="10" nullable=
      "true">VARBINCOL</column>
    <column datatype="VARCHAR" length="10" sourcecodepage="Windows-1252"
      externalfilecodepage="Windows-1252" nullable="true">VCHARCOL</column>
    <column datatype="VARCHAR" length="10" sourcecodepage="Windows-1252"
      externalfilecodepage="Windows-1252" nullable="true">UNIVCHARCOL</column>
  </row>
</table>
```

Bulk Load Configuration File Schema

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

http://www.datadirect.com/ns/bulk/BulkData.xsd

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

Verification of the Bulk Load Configuration File

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

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

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

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

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

```
HDBC      hdbc;
HENV      henv;
void      *driverHandle;
HMODULE   hmod;
PValidateTableFromFile validateTableFromFile;

char      tableName[128];
char      configFile[512];
char      messageList[10240];
SQLLEN    numMessages;

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

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

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

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

```
validateTableFromFile = (PValidateTableFromFile)
  resolveName (hmod, "ValidateTableFromFile");
if (!validateTableFromFile) {
  printf ("Cannot find ValidateTableFromFile!\n");
  exit (255);
}

messageList[0] = 0;
numMessages = 0;

rc = (*validateTableFromFile) (
      driverHandle,
      (const SQLCHAR *) tableName,
      (const SQLCHAR *) configFile,
      (SQLCHAR *) messageList,
      sizeof (messageList),
      &numMessages);
printf ("%d message%s%s\n", numMessages,
       (numMessages == 0) ? "s" :
       ((numMessages == 1) ? " : " : "s : "),
       (numMessages > 0) ? messageList : "");
if (rc == SQL_SUCCESS) {
       printf ("Validate succeeded.\n");
}
else {
    driverError (driverHandle, hmod);
}
```

Refer to "DataDirect Bulk Load Functions" in Chapter 9 of the *DataDirect Connect Series for ODBC Reference* for a complete description of these functions.

Sample Application

DataDirect Tehcnologies provides a sample application that demonstrates the bulk export, verification, and bulk load operations. This application is located in the \example\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.

Determining the Bulk Load Protocol

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

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

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 Chapter 1 "Code Page Values"of the *DataDirect Connect Series for ODBC Reference* is supported for the bulk load data file.

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

The configuration file may optionally define a second code page value for each character column (`externalfilecodepage`). If character data is stored in an external overflow file (see "External Overflow Files" on page 129), this second code page value is used for the external file.

External Overflow Files

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

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

In the case of an external character data file, the character set of the file is governed by the bulk load configuration file. If the bulk load data file is Unicode and the maximum character size of the source data is 1, then the data is stored in its source code page. See "Character Set Conversions" on page 128.

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

Summary of Bulk Load Related Options

Table 3-6 summarizes how DataDirect Bulk Load related connection options work with the drivers. See "Connection Option Descriptions" in each driver chapter for details about configuring the options.

Table 3-6. Summary: Bulk Load Connection Options

Option	Characteristic
Batch Size	An integer value that specifies the number of rows at a time that the driver sends to the database during bulk operations.
Bulk Binary Threshold	An integer value that specifies the maximum size, in KB, of binary data exported to the bulk data file. Any data exceeding this size is exported to an external file.
Bulk Character Threshold	An integer value that specifies the maximum size, in KB, of character data exported to the bulk data file. Any data exceeding this size is exported to an external file.
Enable Bulk Load	When enabled, the driver uses database bulk load protocols. When not enabled, the driver uses standard parameter arrays.

4 Configuring the Product on UNIX/Linux

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

- "Environment Variables" on page 131
- "The Test Loading Tool" on page 136
- "Data Source Configuration" on page 137
- "The demoodbc Application" on page 160
- "The example Application" on page 160
- "DSN-less Connections" on page 161
- "File Data Sources" on page 167
- "UTF-16 Applications on UNIX and Linux" on page 168

See "Environment-Specific Information" on page 59 for additional platform information.

Environment Variables

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

Library Search Path

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

```
echo $SHELL
```

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

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

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

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

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

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

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

```
env
```

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

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

ODBCINI

Setup installs in the product installation directory a default system information file, named odbc.ini, that contains data sources. See "Data Source Configuration" on page 137 for an explanation of the odbc.ini file. The system administrator can choose to rename the file and/or move it to another location. In either case, the environment variable ODBCINI must be set to point to the fully qualified path name of the odbc.ini file.

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

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

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

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

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

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

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

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

ODBCINST

Setup installs in the product installation directory a default file, named odbcinst.ini, for use with DSN-less connections. See "DSN-less Connections" on page 161 for an explanation of the odbcinst.ini file. The system administrator can choose to rename the file or move it to another location. In either case, the environment variable ODBCINST must be set to point to the fully qualified path name of the odbcinst.ini file.

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

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

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

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

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

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

1 The driver checks the ODBCINST variable

2 The driver checks $HOME for .odbcinst.ini

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

DD_INSTALLDIR

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

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

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

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

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

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

1. The driver checks the DD_INSTALLDIR variable
2. The driver checks the odbc.ini or the odbcinst.ini files for the InstallDir keyword (see "Configuration Through the odbc.ini File" on page 141 for a description of the InstallDir keyword)

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

The next step is to test load the driver.

The Test Loading Tool

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

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

For example, if the drivers are installed in /opt/odbc/lib, the following command attempts to load the 32-bit Oracle Wire Protocol 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 tool. The HP-UX version, however, requires the full path.

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

See "Version String Information" on page 74 for details about version strings.

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

Data Source Configuration

In the UNIX and Linux environments, a system information file is used to store data source information. Setup installs a default version of this file, called odbc.ini, in the product installation directory (see "ODBCINI" on page 133 for details about relocating and renaming this file). This is a plain text file that contains data source definitions.

If you have a Motif graphical user interface (GUI) in your UNIX or Linux environment, you can use the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) to create or modify data source definitions in this file (see "Configuration Through the Administrator" on page 138 for details). On Linux, you can determine if you are using Motif through the following command:

```
rpm -qa |grep motif
```

The rpm command returns output similar to:

```
nc-linuxqa3[/home2/users/mike] rpm -qa |grep motif
openmotif-2.2.2-124
openmotif-devel-2.2.2-124
```

Commands for determining whether Motif is installed vary among UNIX systems. Please refer to your operating system documentation for details specific to your system.

If you are not using a GUI, you can use any text editor to create or modify data source definitions directly in the odbc.ini file. See "Configuration Through the odbc.ini File" on page 141 for details.

Configuration Through the Administrator

The UNIX ODBC Administrator is located in the /tools directory of the product installation directory. For example,

```
/opt/odbc/tools/odbcadmin
```

To configure a data source:

1 To start the UNIX ODBC Administrator, change to the *install_dir*/tools directory, where *install_dir* is the path to the product installation directory. At a command prompt, enter:

- UNIX: `odbcadmin`
- Linux: ./`odbcadmin`

The Administrator dialog box appears.

2 Click either the **User 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 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.

- **File DSN**: If you are configuring an existing file data source, select the appropriate data source file and click **Configure** to display the driver 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 Directory**. The next time that you open the Administrator, it displays data source files from this directory.

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

See the individual driver chapters for specific information about the driver Setup dialog. See Chapter 1 “Quick Start Connect” on page 41 for an explanation of different types of data sources.

Drivers

The Drivers tab shows a list of all installed ODBC drivers.

Tracing

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

You enable tracing by selecting the **Enable Tracing** check box on the Tracing tab of the Administrator. Clear the check box to disable tracing. 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.

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.

DataDirect Technologies ships a default shared object, odbctrac.so, to perform tracing. If you want to use a custom shared object instead, type the path and name of the shared object in the Trace Library field or click **Browse** to select a shared object.

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

For a more complete discussion of tracing, refer to "ODBC Trace" in Chapter 1 of the *DataDirect Connect Series for ODBC Troubleshooting Guide*.

When you are finished with the UNIX ODBC Administrator, click **OK** or **Cancel**. If you click **OK**, any changes you have made to the Trace tab are accepted and the Administrator closes. If you click **Cancel**, the Administrator closes without saving any changes.

Configuration Through the odbc.ini 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 2=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. Descriptions of these attributes are in each individual driver chapter. See "Sample Default odbc.ini File" on page 144 for sample data sources.

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

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

This keyword defines the path of the default location for file data sources (see "File Data Sources" on page 167).

NOTE: This section is not included in the default odbc.ini that is installed by the product installer. You can add this section manually or, if you are using the UNIX ODBC Administrator, it will be added automatically when you click **Set Directory** on the File DSN tab (see Step 2 under "Configuration Through the Administrator" on page 138).

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

```
[ODBC]
IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/odbctrac.so
```

The IANAAppCodePage keyword defines the default value that all UNIX/Linux drivers use if individual data sources have not specified a different value. See the individual driver chapters and refer to Chapter 1 "Code Page Values" in the *DataDirect Connect Series for ODBC Reference* 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.

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 or `[ODBC]` section exists, they check for an `[ODBC]` section in the odbcinst.ini file. See "DSN-less Connections" on page 161 for details.

Tracing

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

The [ODBC] section of the system information file includes three keywords related to tracing: Trace, TraceFile, and TraceDll. For example:

```
Trace=1
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/odbctrac.so
```

In this example, tracing is enabled, trace information is logged in a file named odbctrace.out, and odbctrac.so performs the tracing.

You enable tracing by setting the value of Trace to 1. Set the value to 0 to disable tracing. 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.

To specify the path and name of the trace log file, enter it as the value for TraceFile. If no location is specified, the trace log resides in the working directory of the application you are using.

DataDirect Technologies ships a default shared object, odbctrac.so, to perform tracing. If you want to use a custom shared object instead, enter the path and name of the shared object as the value for TraceDll.

For a more complete discussion of tracing, refer to "ODBC Trace" in Chapter 1 of the *DataDirect Connect Series for ODBC Troubleshooting Guide*.

Sample Default odbc.ini File

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

```
[ODBC Data Sources]
DB2 Wire Protocol=DataDirect 6.0 DB2 Wire Protocol
dBASE=DataDirect 6.0 dBASEFile(*.dbf)
FoxPro3=DataDirect 6.0 dBASEFile(*.dbf)
Greenplum Wire Protocol=DataDirect 6.0 Greenplum Wire Protocol
Informix Wire Protocol=DataDirect 6.0 Informix Wire Protocol
MySQL Wire Protocol=DataDirect 6.0 MySQL Wire Protocol
Oracle Wire Protocol=DataDirect 6.0 Oracle Wire Protocol
PostgreSQL Wire Protocol=DataDirect 6.0 PostgreSQL Wire Protocol
SQLServ Native Wire Protocol=DataDirect 6.0 SQL Server Native Wire Protocol
Sybase Wire Protocol=DataDirect 6.0 Sybase Wire Protocol
Teradata=DataDirect 6.0 Teradata
Text=DataDirect 6.0 TextFile(*.*)
Informix=DataDirect 6.0 Informix
Oracle=DataDirect 6.0 Oracle
SQLServ Classic Wire Protocol=DataDirect 6.0 SQL Server Classic Wire Protocol

[DB2 Wire Protocol]
Driver=ODBCHOME/lib/ivdb224.so
Description=DataDirect 6.0 DB2 Wire Protocol
AccountingInfo=
AddStringToCreateTable=
AlternateID=
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=0
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
CatalogSchema=
CharsetFor65535=0
ClientHostName=
ClientUser=
#Collection applies to z/OS and iSeries only
Collection=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CurrentFuncPath=
#Database applies to DB2 UDB only
```

```
Database=<database_name>
DefaultIsolationLevel=1
DynamicSections=200
EnableBulkLoad=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
GrantAuthid=PUBLIC
GrantExecute=1
GSSClient=native
HostNameInCertificate=
IpAddress=<DB2_server_host>
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
#Location applies to z/OS and iSeries only
Location=<location_name>
LogonID=
MaxPoolSize=100
MinPoolSize=0
Password=
PackageCollection=NULLID
PackageNamePrefix=DD
PackageOwner=
Pooling=0
ProgramID=
QueryTimeout=0
ReportCodePageConversionErrors=0
TcpPort=50000
TrustStore=
TrustStorePassword=
UseCurrentSchema=1
ValidateServerCertificate=1
WithHold=1
XMLDescribeType=-10

[dBASE]
Driver=ODBCHOME/lib/ivdbf24.so
```

```
Description=DataDirect 6.0 dBASEFile(*.dbf)
ApplicationUsingThreads=1
CacheSize=4
CreateType=dBASE5
Database=ODBCHOME/demo
DataFileExtension=DBF
ExtensionCase=UPPER
FileOpenCache=0
IntlSort=0
LockCompatibility=dBASE
Locking=RECORD
UseLongNames=0
UseLongQualifiers=0

[FoxPro3]
Driver=ODBCHOME/lib/ivdbf24.so
Description=DataDirect 6.0 dBASEFile(*.dbf)
ApplicationUsingThreads=1
CacheSize=4
CreateType=FoxPro30
Database=ODBCHOME/demo
DataFileExtension=DBF
ExtensionCase=UPPER
FileOpenCache=0
IntlSort=0
LockCompatibility=Fox
Locking=RECORD
UseLongNames=0
UseLongQualifiers=0

[Greenplum Wire Protocol]
Driver=ODBCHOME/lib/ivgplm24.so
Description=DataDirect 6.0 Greenplum Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
DefaultLongDataBuffLen=2048
EnableDescribeParam=0
```

```
EncryptionMethod=0
ExtendedColumnMetaData=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
FetchTWFSasTime=0
HostName=<Greenplum_host>
HostNameInCertificate=
InitializationString=
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
Password=
Pooling=0
PortNumber=<Greenplum_server_port>
QueryTimeout=0
ReportCodepageConversionErrors=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
XMLDescribeType=-10

[Informix Wire Protocol]
Driver=ODBCHOME/lib/ivifcl24.so
Description=DataDirect 6.0 Informix Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
CancelDetectInterval=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
HostName=<Informix_host>
LoadBalancing=0
LogonID=
```

```
Password=
PortNumber=<Informix_server_port>
ServerName=<Informix_server>
TrimBlankFromIndexName=1
UseDelimitedIdentifiers=0

[MySQL Wire Protocol]
Driver=ODBCHOME/lib/ivmysql24.so
Description=DataDirect 6.0 MySQL Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
DefaultLongDataBuffLen=1024
EnableDescribeParam=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
HostName=<MySQL_host>
HostNameInCertificate=
InteractiveClient=0
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
Password=
MaxPoolSize=100
MinPoolSize=0
Pooling=0
PortNumber=<MySQL_server_port>
QueryTimeout=0
ReportCodepageConversionErrors=0
TreatBinaryAsChar=0
TrustStore=
TrustStorePassword=
```

```
ValidateServerCertificate=1

[Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora24.so
Description=DataDirect 6.0 Oracle Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
ArraySize=60000
AuthenticationMethod=1
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
CachedCursorLimit=32
CachedDescLimit=0
CatalogIncludesSynonyms=1
CatalogOptions=0
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
DefaultLongDataBuffLen=1024
DescribeAtPrepare=0
EnableBulkLoad=0
EnableDescribeParam=0
EnableNcharSupport=0
EnableScrollableCursors=1
EnableStaticCursorsForLongData=0
EnableTimestampWithTimeZone=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
GSSClient=native
HostName=<Oracle_server>
HostNameInCertificate=
InitializationString=
KeyPassword=
KeyStore=
KeyStorePassword
LoadBalanceTimeout=0
LoadBalancing=0
```

```
LocalTimeZoneOffset=
LockTimeOut=-1
LoginTimeout=15
LogonID=
Password=
MaxPoolSize=100
MinPoolSize=0
Pooling=0
PortNumber=<Oracle_server_port>
ProcedureRetResults=0
QueryTimeout=0
ReportCodePageConversionErrors=0
ReportRecycleBin=0
ServerName=<server_name_in_tnsnames.ora>
ServerType=0
ServiceName=
SID=<Oracle_System_Identifier>
TimestampeEscapeMapping=0
TNSNamesFile=<tnsnames.ora_filename>
TrustStore=
TrustStorePassword=
UseCurrentSchema=1
ValidateServerCertificate=1
WireProtocolMode=1

[PostgreSQL Wire Protocol]
Driver=ODBCHOME/lib/ivpsql24.so
Description=DataDirect 6.0 PostgreSQL Wire Protocol
AlternateServers=
ApplicationUsingThreads=1
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
DefaultLongDataBuffLen=2048
EnableDescribeParam=0
EncryptionMethod=0
ExtendedColumnMetaData=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
```

```
FetchTSWTZasTimestamp=0
FetchTWFSasTime=0
HostName=<PostgreSQL_host>
HostNameInCertificate=
InitializationString=
KeyPassword=
KeyStore=
KeyStorePassword=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
Password=
Pooling=0
PortNumber=<PostgreSQL_server_port>
QueryTimeout=0
ReportCodepageConversionErrors=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
XMLDescribeType=-10

[SQLServ Native Wire Protocol]
Driver=ODBCHOME/lib/ivsqls24.so
Description=DataDirect 6.0 SQL Server Native Wire Protocol
AlternateServers=
Always Report Trigger Results=0
AnsiNPW=1
ApplicationName=
ApplicationUsingThreads=1
AuthenticationMethod=1
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
BulkLoadOptions=2
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
```

```
DefaultLongDataBuffLen=1024
EnableBulkLoad=0
EnableQuotedIdentifiers=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
FetchTSWTZasTimestamp=0
FetchTWFSasTime=1
GSSClient=native
HostName=<SQL_Server_host>
HostNameInCertificate=
InitializationString=
Language=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
PacketSize=-1
Password=
Pooling=0
PortNumber=<SQL_Server_server_port>
QueryTimeout=0
ReportCodePageConversionErrors=0
SnapshotSerializable=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
WorkStationID=
XML Describe Type=-10

[Sybase Wire Protocol]
Driver=ODBCHOME/lib/ivase24.so
Description=DataDirect 6.0 Sybase Wire Protocol
AlternateServers=
ApplicationName=
ApplicationUsingThreads=1
ArraySize=50
AuthenticationMethod=0
```

```
BulkBinaryThreshold=32
BulkCharacterThreshold=-1
BulkLoadBatchSize=1024
Charset=
ConnectionReset=0
ConnectionRetryCount=0
ConnectionRetryDelay=3
CursorCacheSize=1
Database=<database_name>
DefaultLongDataBuffLen=1024
EnableBulkLoad=0
EnableDescribeParam=0
EnableQuotedIdentifiers=0
EncryptionMethod=0
FailoverGranularity=0
FailoverMode=0
FailoverPreconnect=0
GSSClient=native
HostNameInCertificate=
InitializationString=
Language=
LoadBalanceTimeout=0
LoadBalancing=0
LoginTimeout=15
LogonID=
MaxPoolSize=100
MinPoolSize=0
NetworkAddress=<Sybase_host,Sybase_server_port>
OptimizePrepare=1
PacketSize=0
Password=
Pooling=0
QueryTimeout=0
RaiseErrorPositionBehavior=0
ReportCodePageConversionErrors=0
SelectMethod=0
ServicePrincipalName=
TruncateTimeTypeFractions=0
TrustStore=
TrustStorePassword=
ValidateServerCertificate=1
```

```
WorkStationID=

[Teradata]
Driver=ODBCHOME/lib/ivtera24.so
Description=DataDirect 6.0 Teradata
AccountString=
AuthenticationDomain=
AuthenticationPassword=
AuthenticationUserid=
CharacterSet=ASCII
DBCName=<Teradata_server>
Database=
EnableDataEncryption=0
EnableExtendedStmtInfo=0
EnableLOBs=1
EnableReconnect=0
IntegratedSecurity=0
LoginTimeout=20
LogonID=
MapCallEscapeToExec=0
MaxRespSize=8192
Password=
PortNumber=1025
PrintOption=N
ProcedureWithSplSource=Y
ReportCodePageConversionErrors=0
SecurityMechanism=
SecurityParameter=
ShowSelectableTables=1
TDProfile=
TDRole=
TDUserName=

[Text]
Driver=ODBCHOME/lib/ivtxt24.so
Description=DataDirect 6.0 TextFile(*.*)
AllowUpdateAndDelete=0
ApplicationUsingThreads=1
CacheSize=4
CenturyBoundary=20
Database=ODBCHOME/demo
```

```
DataFileExtension=TXT
DecimalSymbol=.
Delimiter=
FileOpenCache=0
FirstLineNames=0
IntlSort=0
ScanRows=25
TableType=Comma
UndefinedTable=GUESS

[Informix]
Driver=ODBCHOME/lib/ivinf24.so
Description=DataDirect 6.0 Informix
ApplicationUsingThreads=1
CancelDetectInterval=0
CursorBehavior=0
Database=<database_name>
EnableInsertCursors=0
GetDBListFromInformix=1
HostName=<Informix_host>
LogonID=
Password=
Protocol=onsoctcp
ServerName=<Informix_server>
Service=<Informix_service_name>
TrimBlankFromIndexName=1

[Oracle]
Driver=ODBCHOME/lib/ivor824.so
Description=DataDirect 6.0 Oracle
AlternateServers=
ApplicationUsingThreads=1
ArraySize=60000
CatalogIncludesSynonyms=1
CatalogOptions=0
ClientVersion=9iR2
ConnectionRetryCount=0
ConnectionRetryDelay=3
DefaultLongDataBuffLen=1024
DescribeAtPrepare=0
EnableDescribeParam=0
```

```
EnableNcharSupport=0
EnableScrollableCursors=1
EnableStaticCursorsForLongData=0
EnableTimestampWithTimeZone=0
LoadBalancing=0
LocalTimeZoneOffset=
LockTimeOut=-1
LogonID=
Password=
OptimizeLongPerformance=0
ProcedureRetResults=0
ReportCodePageConversionErrors=0
ReportRecycleBin=0
ServerName=<Oracle_server>
TimestampEscapeMapping=0
UseCurrentSchema=1

[SQLServ Classic Wire Protocol]
Driver=ODBCHOME/lib/ivmsss24.so
Description=DataDirect 6.0 SQL Server Classic Wire Protocol
Address=<SQLServer_host, SQLServer_server_port>
AlternateServers=
AnsiNPW=Yes
ConnectionRetryCount=0
ConnectionRetryDelay=3
Database=<database_name>
FetchTSWTZasTimestamp=0
FetchTWFSasTime=0
LoadBalancing=0
LogonID=
Password=
QuotedId=No
ReportCodepageConversionErrors=0
SnapshotSerializable=0

[ODBC]
IANAAppCodePage=4
InstallDir=ODBCHOME
Trace=0
TraceFile=odbctrace.out
TraceDll=ODBCHOME/lib/odbctrac.so
```

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

To modify a data source:

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

   Consult the "Connection String Attributes" table of each driver chapter for other specific attribute values.
3. After making all modifications, save the odbc.ini file and close the text editor.

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

To create a new data source:

1. Using a text editor, open the odbc.ini file.
2. Copy an appropriate existing default data source definition and paste it to another location in the file.
3. 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]`.
4. Modify the attributes in the new definition as necessary based on your system specifics, for example, enter the host name and port number of your system in the appropriate location.

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

5 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.

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

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

Translators

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

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

For example, the 32-bit DB2 driver would be:

```
[DB2]
Driver=ODBCHOME/lib/ivdb224.so
Description=DataDirect 6.0 DB2 Wire Protocol
TranslationSharedLibrary=ODBCHOME/lib/ivtrn24.so
```

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

The final step is to verify that the driver connects and passes SQL.

The demoodbc Application

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

```
demoodbc -uid user_name -pwd password data_source_name
```

For example:

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

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

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

The example Application

DataDirect Technologies ships an application, named example, that is installed in the example subdirectory in 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 readme 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.

Setup installs a default version of this file in the product installation directory (see "ODBCINST" on page 134 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 Oracle Wire Protocol=Installed.
```

This section also includes additional information for each driver.

The next section of the file is named `[Administrator]`. The keyword in this section, `AdminHelpRootDirectory`, is required for the UNIX ODBC Administrator to locate its help system. The installation process automatically provides the correct value for this keyword.

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 odbc.ini File" on page 141 for a description of the other keywords this section.

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

Sample odbcinst.ini File

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

```
[ODBC Drivers]
DataDirect 6.0 DB2 Wire Protocol=Installed
DataDirect 6.0 dBASEFile (*.dbf)=Installed
DataDirect 6.0 Greenplum Wire Protocol=Installed
DataDirect 6.0 Informix Wire Protocol=Installed
DataDirect 6.0 MySQL Wire Protocol=Installed
DataDirect 6.0 Oracle Wire Protocol=Installed
DataDirect 6.0 PostgreSQL Wire Protocol=Installed
DataDirect 6.0 SQL Server Native Wire Protocol=Installed
DataDirect 6.0 Sybase Wire Protocol=Installed
DataDirect 6.0 Teradata=Installed
DataDirect 6.0 TextFile (*.*)=Installed
DataDirect 6.0 Informix=Installed
DataDirect 6.0 Oracle=Installed
DataDirect 6.0 SQL Server Classic Wire Protocol=Installed

[DataDirect 6.0 DB2 Wire Protocol]
Driver=ODBCHOME/lib/ivdb224.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivdb224.so
SQLevel=0

[DataDirect 6.0 dBASEFile (*.dbf)]
Driver=ODBCHOME/lib/ivdbf24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileExtns=*.dbf
FileUsage=1
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivdbf24.so
SQLLevel=0
```

```
[DataDirect 6.0 Greenplum Wire Protocol]
Driver=ODBCHOME/lib/ivgplm24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivgplm24.so
SQLLevel=0

[DataDirect 6.0 Informix Wire Protocol]
Driver=ODBCHOME/lib/ivifcl24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivifcl24.so
SQLLevel=0

[DataDirect 6.0 MySQL Wire Protocol]
Driver=ODBCHOME/lib/ivmysql24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivmysql24.so
SQLLevel=0

[DataDirect 6.0 Oracle Wire Protocol]
Driver=ODBCHOME/lib/ivora24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivora24.so
SQLLevel=0
```

```
[DataDirect 6.0 PostgreSQL Wire Protocol]
Driver=ODBCHOME/lib/ivpsql24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivpsql24.so
SQLLevel=0

[DataDirect 6.0 SQL Server Native Wire Protocol]
Driver=ODBCHOME/lib/ivsqls24.so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/isqls24s.so
SQLLevel=0

[DataDirect 6.0 Sybase Wire Protocol]
Driver=ODBCHOME/lib/ivase24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivase24.so
SQLLevel=0

[DataDirect 6.0 Teradata]
Driver=ODBCHOME/lib/ivtera24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivtera24s.so
SQLLevel=0
```

```
[DataDirect 6.0 TextFile (*.*)]
Driver=ODBCHOME/lib/ivtxt24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileExtns=*.*
FileUsage=1
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivtxt24.so
SQLLevel=0

[DataDirect 6.0 Informix]
Driver=ODBCHOME/lib/ivinf24.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivinf24s.so
SQLLevel=0

[DataDirect 6.0 Oracle]
Driver=ODBCHOME/lib/ivor824.so
APILevel=0
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivor824s.so
SQLLevel=0

[DataDirect 6.0 SQL Server Classic Wire Protocol]
Driver=ODBCHOME/lib/ivmsss24.so
APILevel=2
ConnectFunctions=YYY
DriverODBCVer=3.52
FileUsage=0
HelpRootDirectory=ODBCHOME/help
Setup=ODBCHOME/lib/ivmsss24s.so
SQLLevel=0
```

```
[Administrator]
HelpRootDirectory=ODBCHOME/adminhelp

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

File Data Sources

The Driver Manager on UNIX and Linux supports file data sources. The advantage of a file data source is that it can be stored on a server and accessed by other machines, either Windows, UNIX, or Linux. See "Quick Start Connect" on page 41 for a general description of ODBC data sources on both Windows and UNIX.

A file data source is simply a text file that contains connection information. It can be created through the UNIX ODBC Administrator (see "Configuration Through the Administrator" on page 138) or it can be created with a text editor. The file normally has an extension of .dsn.

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

```
[ODBC]
Driver=DataDirect Oracle Wire Protocol
Port=1522
HostName=ORA2
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" on page 161).

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\Oraclewp.dsn
```

or, on UNIX and Linux:

```
FILEDSN=/home/users/john/filedsn/Oraclewp2.dsn
```

If no path is specified for the file data source, the Driver Manager uses the `[ODBC File DSN]` setting in the odbc.ini file to locate file data sources (see "Configuration Through the odbc.ini File" on page 141 for details). If the `[ODBC File DSN]` setting is not defined, the Driver Manager uses the installation directory setting in 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/Oraclewp2.dsn;UID=james;PWD=test01
```

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 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);
```

Part 2: The 32-Bit/64-Bit Drivers

This part describes the drivers that are available in both 32- and 64-bit versions. See "Part 3: The 32-Bit Drivers" on page 831 for the drivers that are available only in 32-bit versions.

This part contains the following chapters:

- Chapter 5 "The DB2 Wire Protocol Driver" on page 173
- Chapter 6 "The Greenplum Wire Protocol Driver" on page 263
- Chapter 7 "The Informix Wire Protocol Driver" on page 309
- Chapter 8 "The MySQL Wire Protocol Driver" on page 337
- Chapter 9 "The Oracle Wire Protocol Driver" on page 385
- Chapter 10 "The PostgreSQL Wire Protocol Driver" on page 481
- Chapter 11 "The SQL Server Native Wire Protocol Driver" on page 535
- Chapter 12 "The Sybase Wire Protocol Driver" on page 607
- Chapter 13 "The Oracle Driver" on page 695
- Chapter 14 "The SQL Server Classic Wire Protocol Driver" on page 751
- Chapter 15 "The Driver for the Teradata Database" on page 791

5 The DB2 Wire Protocol Driver

The DataDirect Connect Series *for* ODBC DB2 Wire Protocol driver (the DB2 Wire Protocol driver) is available in both 32- and 64-bit versions and supports the following DB2 database servers:

- DB2 V9.1, V9.5, and V9.7 for Linux, UNIX, and Windows via DRDA
- DB2 Universal Database (UDB) v7.*x*, v8.1, and v8.2 for Linux, UNIX, and Windows via DRDA
- DB2 UDB v7.*x*, v8.1, and v9.1 for z/OS via DRDA
- DB2 UDB V5R1, V5R2, V5R3, V5R4, and V6R1 for iSeries

NOTE: This documentation uses the following terms to describe the different DB2 versions:

- "DB2 for Linux/UNIX/Windows" refers to all versions of DB2 for Linux, UNIX, and Windows
- "DB2 for z/OS" refers to all versions of DB2 for z/OS (formerly OS/390)
- "DB2 for iSeries" refers to all versions of DB2 for iSeries (formerly AS/400)

The DB2 Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the DB2 Wire Protocol driver.

Driver Requirements

The server requirement for all platforms is the same. The DB2 database can be installed as the Server Version or the Personal Edition.

The driver has no client requirements.

Binding

The driver does not work properly unless bind packages exist on every server to which you intend to connect.

IMPORTANT: You must have the appropriate privileges for the driver to create and bind packages with your user ID. These privileges are BINDADD for binding packages, CREATEIN on the collection specified by the Package Collection option, and GRANT EXECUTE on the PUBLIC group for executing the packages. These are typically the permissions of a Database Administrator (DBA). If you do not have these privileges, someone who has a user ID with DBA privileges needs to create packages by connecting with the driver.

When connecting for the first time, the driver determines whether bind packages exist on the server. If packages do not exist, the driver creates them automatically using driver data source default values.

NOTE: The initial driver connection to a particular server may take a few minutes because of the number and size of the packages that must be created on the server. Subsequent connections do not incur this delay.

If you change default values in a data source before connecting with the driver for the first time, the new defaults are used when creating the packages. If you want to change these values after the packages have been created, you can re-create the packages with the new values using one of the following methods:

You can create or modify packages from the Modify Bindings tab of the Setup dialog. See Step 6 under "Configuring and Connecting to Data Sources" on page 180 for details.

On UNIX and Linux, you can also create or modify packages through a special bind utility. Depending on the platform of the DB2 server, the minimum attribute values that must be set in the data source to bind packages are:

Linux/UNIX/Windows DB2 Servers: IpAddress, Database, TcpPort

z/OS and iSeries DB2 Servers: IpAddress, Location, TcpPort

Other attribute values also affect binding. See the note for Step 6 under "Configuring and Connecting to Data Sources" on page 180 for details. See "Connection Option Descriptions" on page 208 for a description of these connection string attributes and their values. You must use the default values or specify new ones for these attributes in the DB2 data source section of the odbc.ini file before binding. See Chapter 4 "Configuring the Product on UNIX/Linux" on page 131 for details on creating the DB2 data source.

The bind utility is located in *installation_directory*/bin. After specifying the appropriate connection string attribute values in the odbc.ini file, create or modify packages by entering the command:

```
bindxx dsn
```

where *xx* is the driver level number in the driver file name and *dsn* is the ODBC data source name in the odbc.ini file. You are prompted for a user ID and password if they are not stored in

the data source. If packages are created and bound successfully, a message indicating success appears. If there are problems connecting or creating the packages, an appropriate error message appears.

Creating DB2 Packages Using List Files

You can bind the following list files on your database server to create DB2 packages:

- DDODBC_LUW.lst (DB2 for Linux/UNIX/Windows)
- DDODBC_400.lst (DB2 for iSeries)
- DDODBC_MVS.lst (DB2 for z/OS)

The list files are located in the DB2/bind directory in your DataDirect Connect Series *for* ODBC installation directory. When you bind the list files, if any DataDirect DB2 packages exist, they will be replaced by the new packages. The list files create DB2 packages that, by default, contain 200 dynamic sections and are created in the NULLID collection.

To create DB2 packages by binding list files:

1. Copy the appropriate list (*.lst) file and bind (*.bnd) files located in the /bind directory to a directory on the database server.
2. From the database server directory where you placed the list and bind files, start the DB2 command-line utility. Use the utility to connect to the database where you want to bind the packages. Connect using the following command:

```
connect to database_name user authorization_name using password
```

where:

database_name is the name of the database to which you are connecting.

authorization_name is the name of the user you are authenticating to the server.

password is the user's password.

3 Execute the DB2 bind command:

```
bind @list_file grant public
```

where *list_file* is the name of the list file you want to bind.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Security
- Connection Pooling
- DataDirect Bulk Load

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Kerberos Authentication Requirements

Verify that your environment meets the requirements listed in Table 5-1 before you configure the driver for Kerberos authentication.

Table 5-1. Kerberos Authentication Requirements for the DB2 Wire Protocol Driver

Component	Requirements
Database server	The database server must be running one of the following database versions: ■ DB2 v8.1 or higher for Linux/UNIX/Windows ■ DB2 v7.x or higher for z/OS
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by one of the following methods: ■ Windows Active Directory on one of the following operating systems: • Windows Server 2003 • Windows 2000 Server Service Pack 3 or higher ■ MIT Kerberos 1.4.2 or higher

Workload Manager (WLM)

The Workload Manager (WLM) is a priority and resource manager within DB2 V9.5 for Linux/UNIX/Windows. On z/OS, the WLM is part of the operating system. WLM prioritizes and matches DB2 workloads with available resources. DB2 workloads allow you to categorize similar types of work. For example, a database administrator can create a DB2 workload named Sales to service all connections that come from Sales applications.

The WLM automatically adjusts server resources, such as CPU and memory, based on the service class associated with a DB2 workload. Therefore, an application's performance is tied to the DB2 workload to which it is assigned and, ultimately, to the service class associated with that workload. For example, an application that performs batch work nightly when resource

usage is low can use the default workload. In contrast, sales updates that need to be processed quickly twice a day need to use a workload that is governed by a high priority service class.

It is important to understand that, unless specified otherwise, all work will run in the default workload that is governed by the default service class. To ensure the best performance, consult with your database administrator to verify that your application is associated with the appropriate DB2 workload and service class.

In addition to workload management, WLM also provides monitoring functionality that is useful for troubleshooting. For example, the database administrator can set threshold limits to detect long-running queries and gather information about those queries.

The DB2 Wire Protocol driver allows your application to set client information in the DB2 database that can be used by the WLM to classify work. If you know that your database environment uses WLM, coordinate with your database administrator to determine which connection options you need to set. These options are located on the Advanced tab of the driver Setup dialog.

On DB2 for z/OS, the driver fully supports WLM Classification. All supported DDF work classification attributes are listed in the following table.

Table 5-2. DDF Work Classification Attributes

Attribute	Driver Value/Option	Description
Accounting Info (AI)	Set via AccountingInfo connection option.	Set via connection option.
Correlation Info (CI)	ODBC4DB2	DB2 Connect assigns the application program name by default, but application can set via Client Information APIs.

Table 5-2. DDF Work Classification Attributes (cont.)

Attribute	Driver Value/Option	Description
Collection Name (CN)	The default is NULLID, but this is configurable via the PackageCollection connection option.	Collection name of the first SQL package accessed by the DRDA requester in the unit of work.
Connection Type (CT)	DIST	Always DIST for DDF.
Package Name (PK)	The default initial characters for the ODBC driver packages are DD, but this is configurable via the PackageNamePrefix connection option.	Name of the first DB2 package accessed by the DRDA requester in the unit of Work.
Plan Name (PN)	DISTSERV	Always DISTSERV for DDF.
Procedure Name (PR)	N/A	Name of the procedure called as the first request in the unit of work.
Process Name (PC)	Set via ApplicationName connection option.	Client application name by default, but can be set via connection option.
Subsystem Collection name (SSC)	N/A	Usually the DB2 data sharing group name.
Subsystem Instance (SI)	N/A	MVS subsystem name of the DB2 server.
Sysplex Name (PX)	N/A	Name assigned to sysplex at IPL.
Userid (UI)	USERID specified at connection time.	DDF server thread's primary AUTHID.
Sybsystem Parameter(SPM)	N/A	V8 - assigned the concatenation of client userid/workstation name.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 “Quick Start Connect” on page 41

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

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 205 and Table 5-3 on page 209 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 5-3 on page 209 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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

To configure a DB2 data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

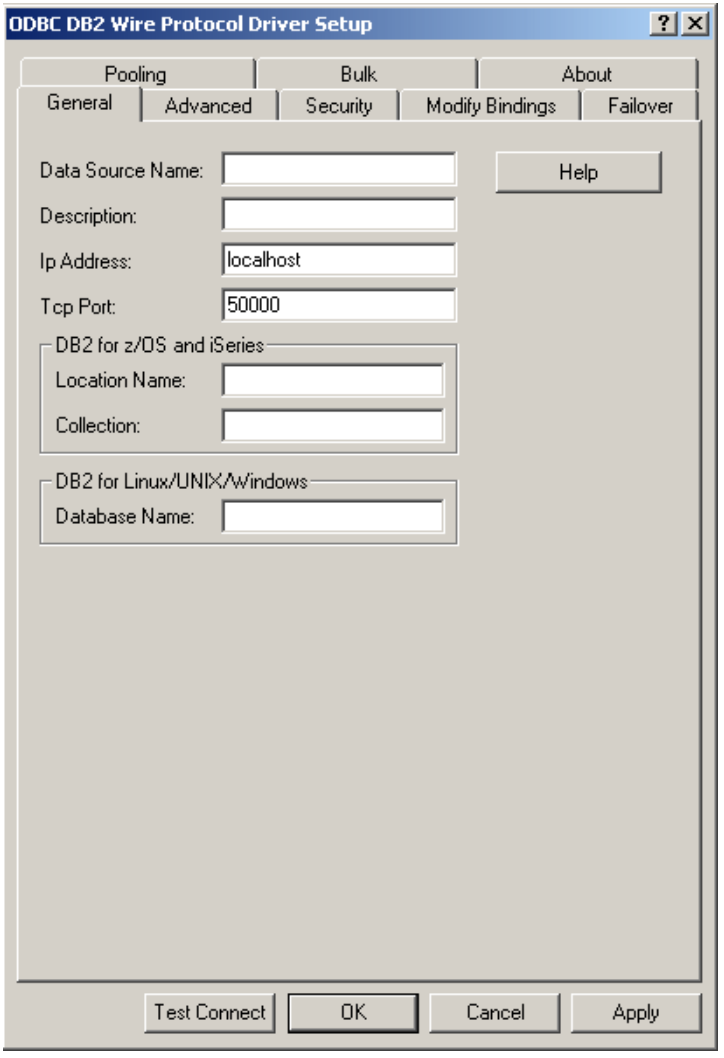
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

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

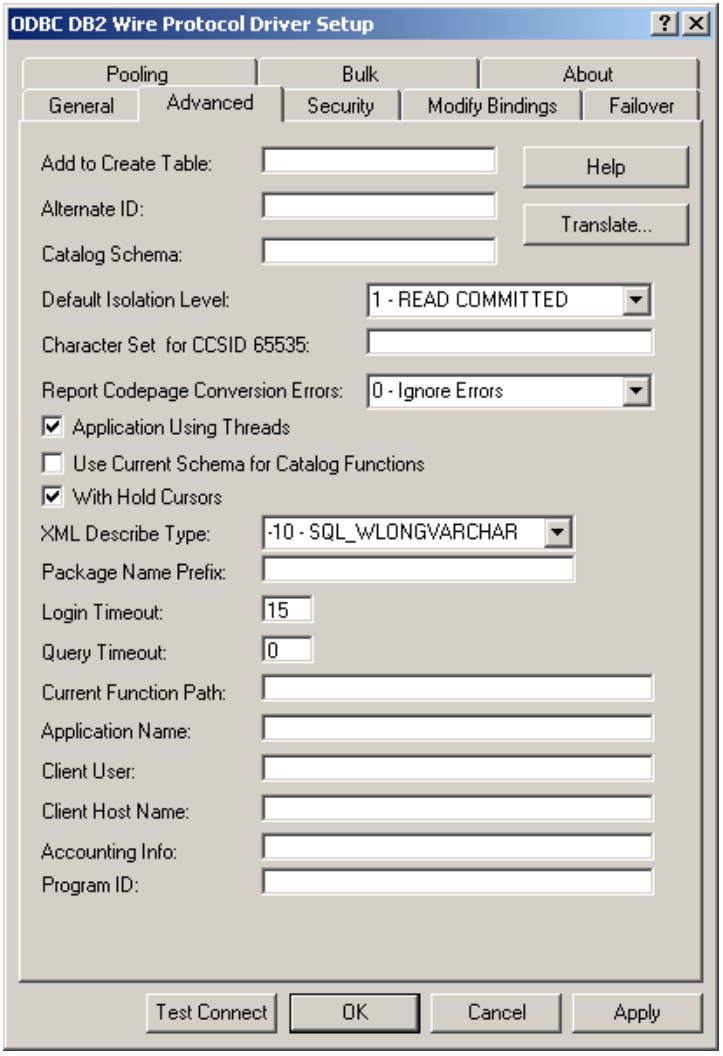


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 224)	None
Description (see page 226)	None
Ip Address (see page 234)	localhost
Tcp Port (see page 245)	50000
Location Name (see page 238)	None
Collection (see page 220)	None
Database Name (see page 224)	None

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



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

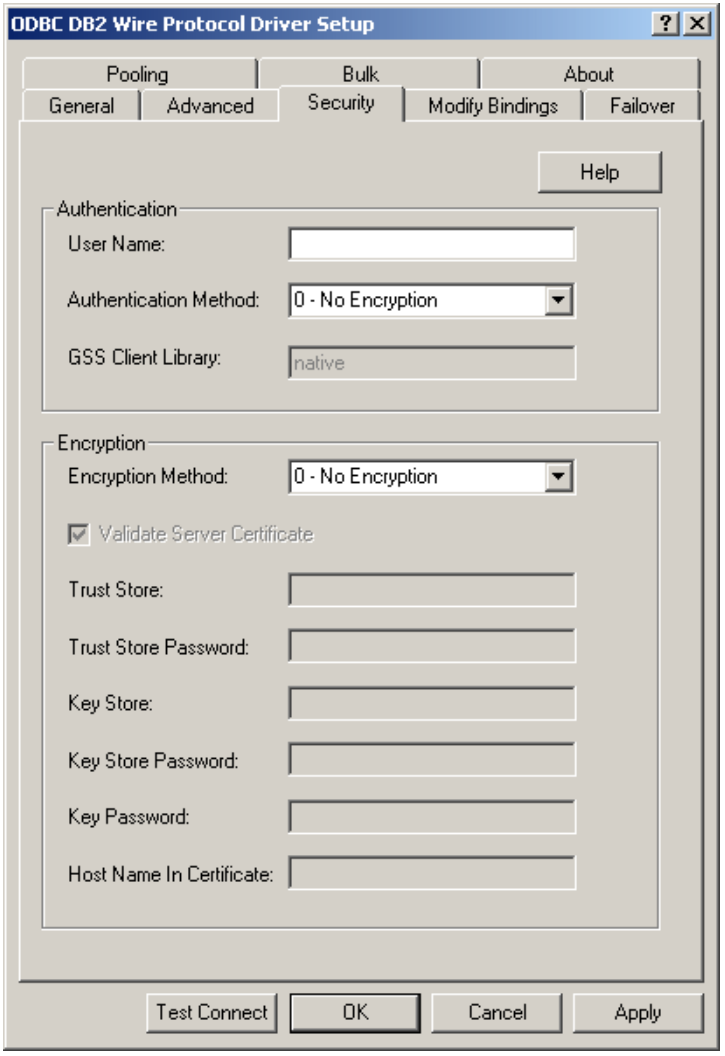
Connection Options: Advanced	Default
Add to Create Table (see page 212)	None

Connection Options: Advanced	Default
Alternate ID (see page 212)	None
Catalog Schema (see page 218)	None
Default Isolation Level (see page 224)	Enabled
Character Set for CCSID 65535 (see page 218)	Disabled
Report Codepage Conversion Errors (see page 244)	Ignore Errors
Application Using Threads (see page 214)	Enabled
Use Current Schema for Catalog Functions (see page 247)	Disabled
With Hold Cursors (see page 249)	Enabled
XML Describe Type (see page 249)	-10
Package Name Prefix (see page 241)	DD
Login Timeout (see page 239)	15
Query Timeout (see page 244)	0
Current Function Path (see page 223)	None
Application Name (see page 214)	None
Client User (see page 219)	None
Client Host Name (see page 219)	None
Accounting Info (see page 211)	None
Program ID (see page 243)	None
IANAAppCodePage (see page 233) UNIX ONLY	4 (ISO 8559-1 Latin 1)

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

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

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

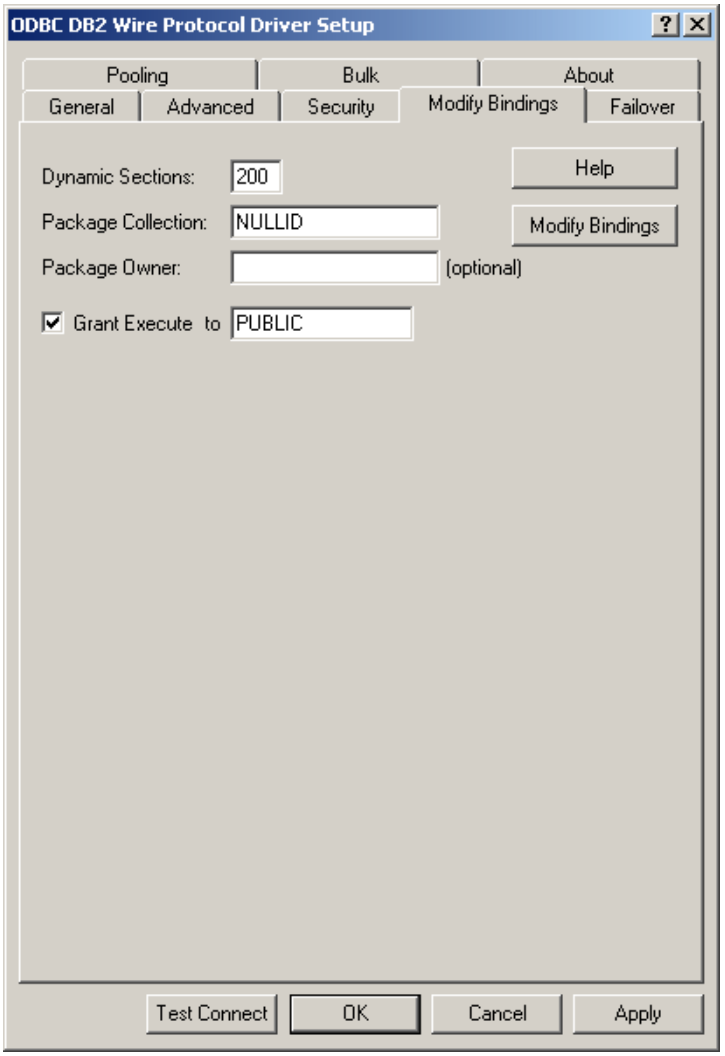


See "Using Security" on page 101 for a general description of authentication and encryption and their configuration requirements.

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

Connection Options: Security	Default
User Name (see page 248)	None
Authentication Method (see page 215)	0 (No Encryption)
GSS Client Library (see page 232)	native
Encryption Method (see page 227)	0 (No Encryption)
Validate Server Certificate (see page 248)	Enabled
Truststore (see page 246)	None
Truststore Password (see page 246)	None
Keystore (see page 235)	None
Keystore Password (see page 236)	None
Key Password (see page 235)	None
Host Name In Certificate (see page 232)	None

6 Optionally, click the **Modify Bindings** tab to configure options for creating or modifying bind packages.



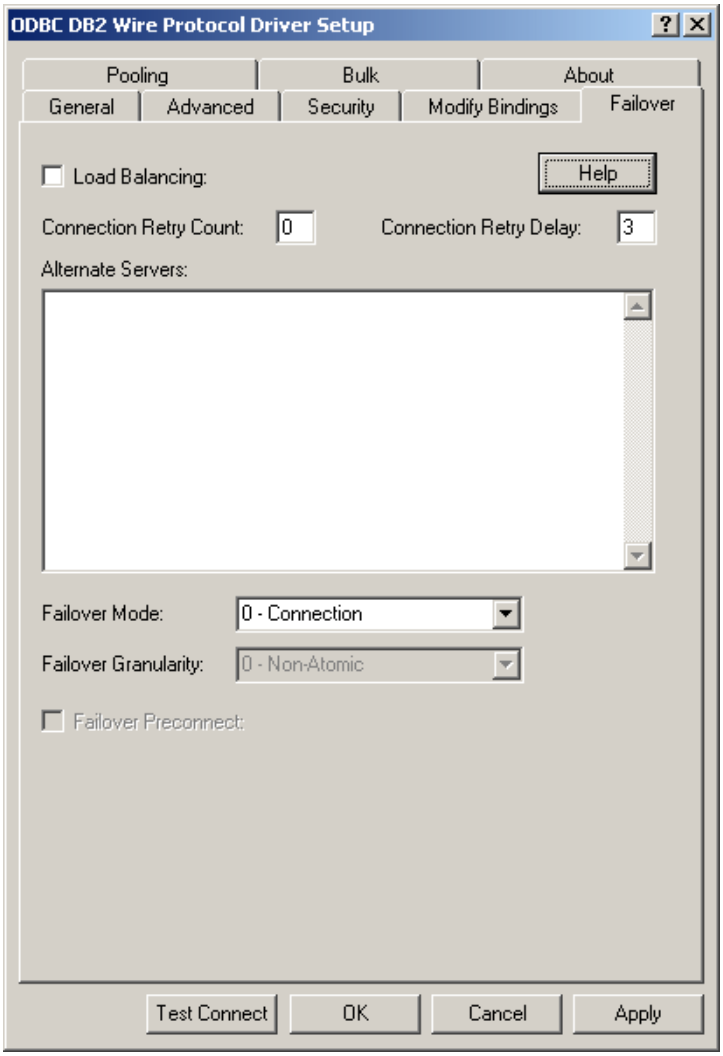
NOTE: If you change any of the values on this tab after having initially created bind packages, you must rebind the packages. The changes are reflected only on new connections after rebinding.

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

Connection Options: Modify Bindings	Default
Dynamic Sections (see page 226)	200
Package Collection (see page 241)	NULLID
Package Owner (see page 242)	None
Grant Execute to [check box] (see page 231)	Enabled
Grant Execute to [field] (see page 231)	PUBLIC

The Modify Bindings tab allows you to create or modify bind packages on the server accessed by the driver. If you connect with the driver before explicitly creating bind packages, the driver creates packages on the server automatically using the current values from the Setup dialog. Alternatively, you can create a bind package before testing the connection. You can also modify bind packages after their creation from the Modify Bindings tab. You must also provide appropriate values for the options on the General tab.

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

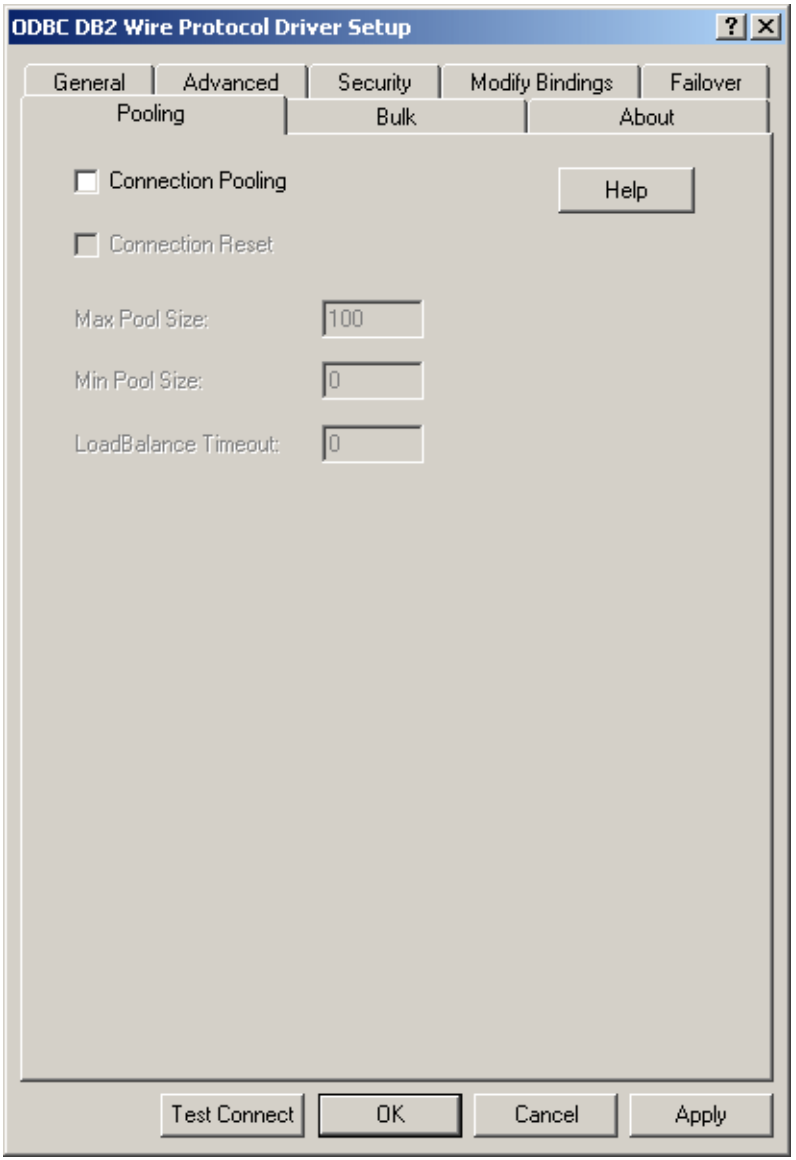


See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 237)	Disabled
Connection Retry Count (see page 222)	0
Connection Retry Delay (see page 222)	3
Alternate Servers (see page 213)	None
Failover Mode (see page 229)	0 (Connection)
Failover Granularity (see page 228)	0 (Non-Atomic)
Failover Preconnect (see page 230)	Disabled

8 Optionally, click the **Pooling** tab to specify connection pooling data source settings.

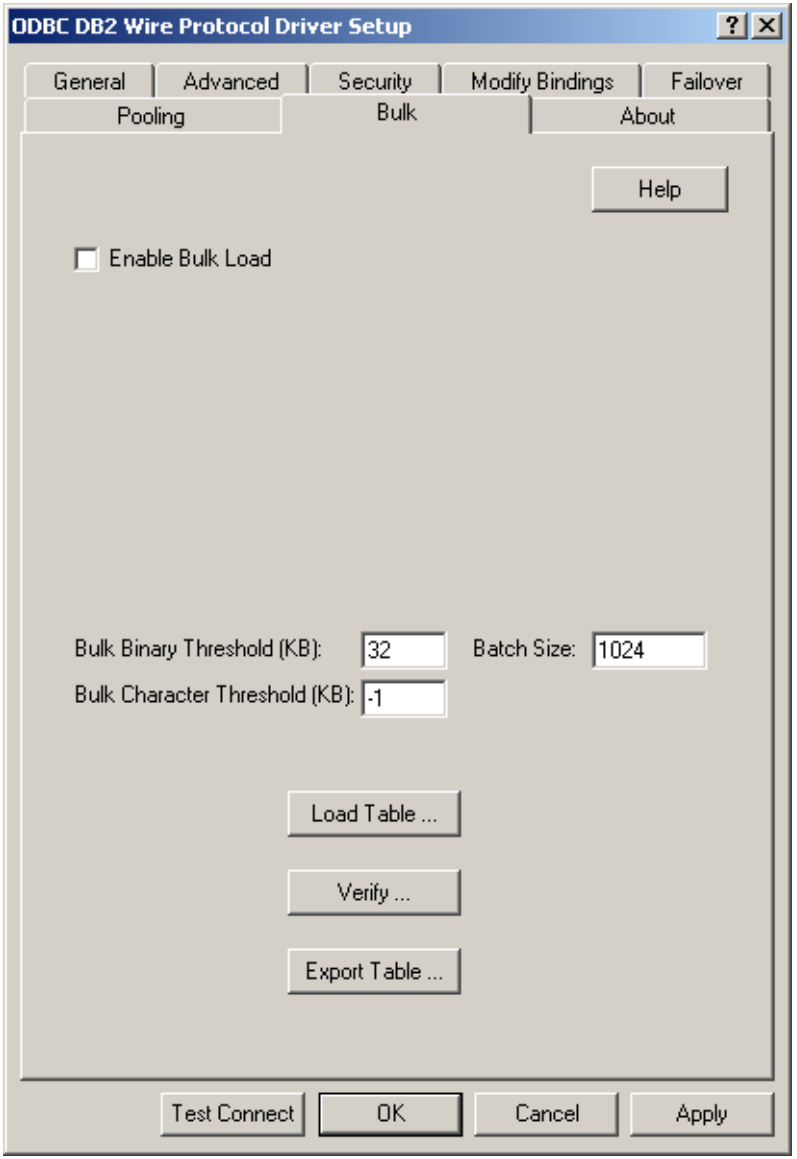


See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 221)	Disabled
Connection Reset (see page 221)	Disabled
Max Pool Size (see page 239)	100
Min Pool Size (see page 240)	0
Load Balance Timeout (see page 237)	0

9 Optionally, click the **Bulk** tab to specify DataDirect Bulk Load data source settings.



See "Using DataDirect Bulk Load" on page 115 for a general description of DataDirect Bulk Load.

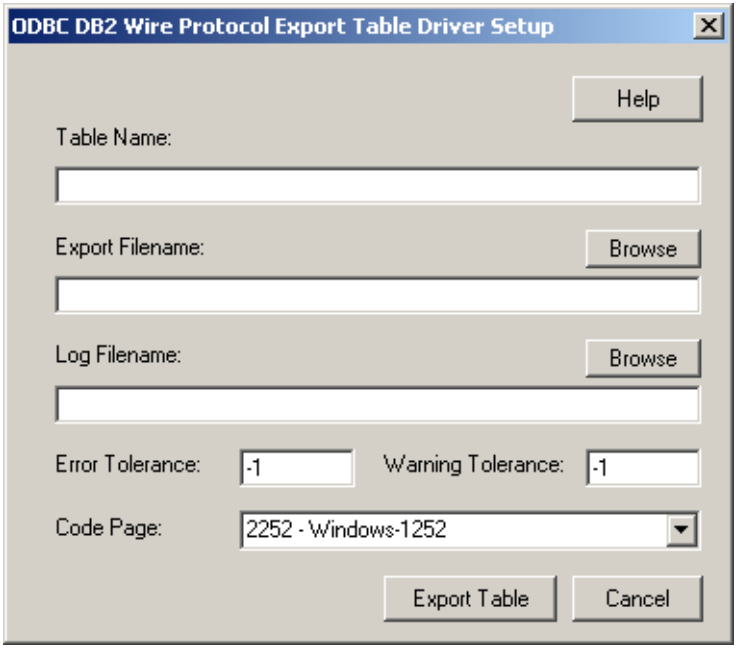
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Bulk	Default
Enable Bulk Load (see page 227)	Disabled
Bulk Binary Threshold (see page 216)	32
Bulk Character Threshold (see page 217)	-1
Batch Size (see page 216)	1024

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

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

10 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.



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" on page 115 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" on page 129 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. This file must not already exist; if the file already exists, an error is returned.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. Events logged to this file are:

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

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

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

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

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

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

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

Code Page: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See "Character Set Conversions" on page 128 for more information.

The default value on Windows is the current code page of the machine. On UNIX/Linux, the default value is 4.

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

11 To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify** from the Bulk tab. See "Verification of the Bulk Load Configuration File" on page 125 for details. The Verify dialog box appears.

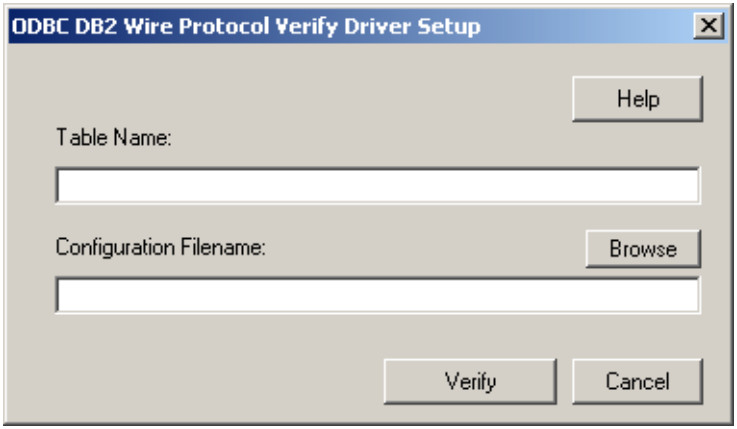
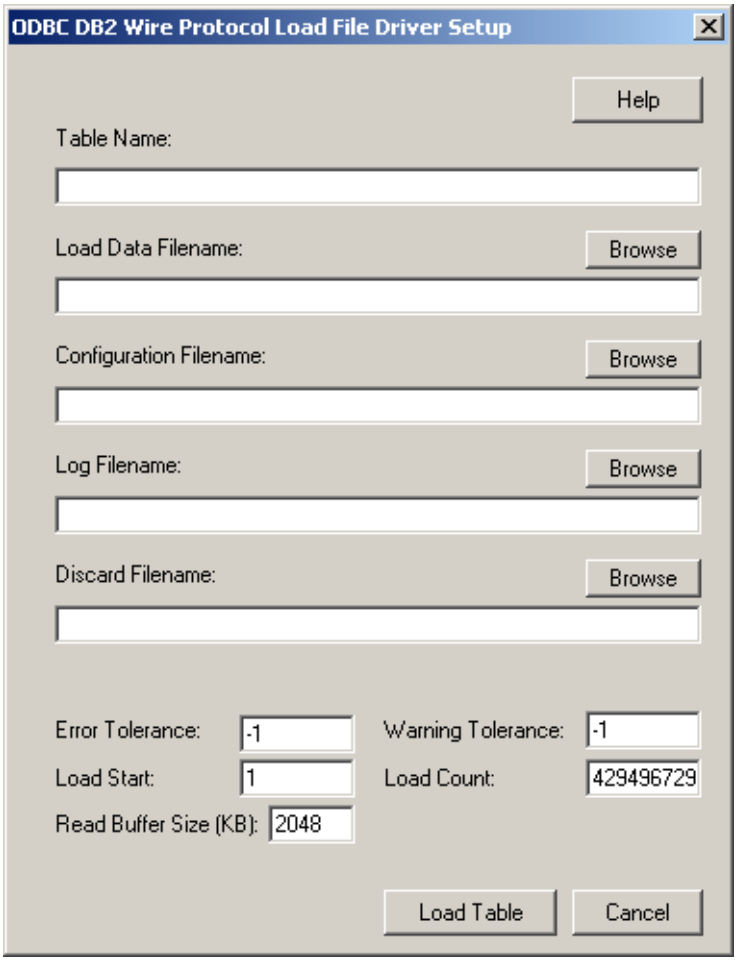


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

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

13 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.



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 to be 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 to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. Supplying 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 supply 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. 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 you do not supply a value for Discard Filename, no discard 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.

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 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 4294967295.

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

14 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" on page 206 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

IMPORTANT: If you have not already created bind packages by clicking the Modify Bindings button on the Modify Bindings tab, the initial connection through the Test Connect button may take a few minutes because of the number and size of the packages that must be created on the server. Subsequent connections occur without this delay.

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

15 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data

source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 5-3 on page 209 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 DB2 for Linux/UNIX/Windows is:

```
DSN=DB2ACCOUNT;DB=DB2DATA;UID=JOHN;PWD=XYZZY
```

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

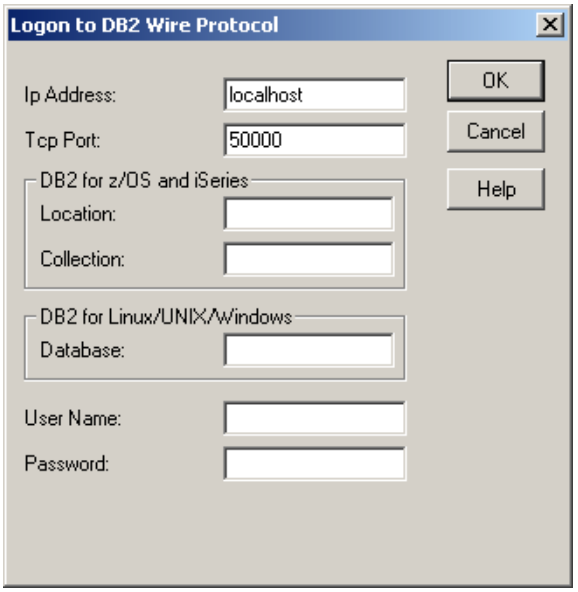
```
FILEDSN=DB2.dsn;DB=DB2DATA;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect DB2 Wire Protocol;
IpAddress=123.456.78.90;PORT=5179;DB=DB2ACCT;UID=JOHN;
PWD=XYZZY
```

Using a Logon Dialog Box

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



In this dialog box, provide the following information:

1 In the Ip Address field, type the IP (Internet Protocol) address of the machine where the catalog tables are stored. Specify the address using the machine's numeric address (for example, `123.456.78.90`) or specify its host name. If you enter a host name, the driver must find this name (with the correct address assignment) in the HOSTS file on the workstation or in a DNS server. The default is localhost.

   The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details concerning these formats.

2 In the Tcp Port field, type the port number that is assigned to the DB2 server on the machine where the catalog tables are stored. Specify either this port's numeric address or its service name (50000 is the default port address). If you specify a service name, the driver must find this name (with the correct port assignment) in the SERVICES file on the workstation.

3 If you are running DB2 for z/OS or iSeries, perform Steps 3a and 3b. Otherwise, skip to Step 4.

   a In the Location field, type the DB2 location name. Use the name defined during the local DB2 installation.

   b By default, the User ID is used for the value of Collection. The User ID must always be used on DB2 for z/OS.

      On iSeries, you can type the name of the schema that is to be the default qualifier for unqualified object names. If you want to access a table outside of this schema, you need to specify the appropriate two-part name, for example, `SELECT * FROM Schema.Tablename`. On iSeries only, Collection is also the current library.

      Skip to Step 5.

4 If you are running DB2 for Linux/UNIX/Windows, type the name of the database to which you want to connect in the Database field.

5 If required, type your logon ID in the User Name field.

6 If required, type your password in the Password field.

7 Click **OK** to complete the logon and to update the values in the Registry.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

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

Table 5-3 lists the connection string attributes supported by the DB2 Wire Protocol driver.

Table 5-3. DB2 Wire Protocol Attribute Names

Attribute (Short Name)	Default
AccountingInfo (AI)	None
AddStringToCreateTable (ASCT)	None
AlternateID (AID)	None
AlternateServers (ASVR)	None
ApplicationName (AN)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
AuthenticationMethod (AM)	0 (No Encryption)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadBatchSize (BLBS)	1024
CatalogSchema (CS)	None
CharsetFor65535 (CF6)	0
ClientHostName (CHN)	None
ClientUser (CU)	None
Collection (COL)	None
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
CurrentFunctionPath (CFP)	None

Table 5-3. DB2 Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
Database (DB)	None
DataSourceName (DSN)	None
DefaultIsolationLevel (DIL)	1 (READ_COMMITTED)
Description (n/a)	None
DynamicSections (DS)	200
EnableBulkLoad (EBL)	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
GrantAuthid (GA)	PUBLIC
GrantExecute (GE)	1 (Enabled)
GSSClientLibrary (GS)	native
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IpAddress (IP)	localhost
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
Location (LOC)	None
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
PackageCollection (PC)	NULLID
PackageNamePrefix (PNP)	DD
PackageOwner (PO)	None

Table 5-3. DB2 Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
Password (PWD)	None
Pooling (POOL)	0 (disabled)
ProgramID (PID)	None
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
TcpPort (PORT)	50000
Truststore (TS)	None
TruststorePassword (TSP)	None
UseCurrentSchemaforCatalogFunctions (UCS)	0 (disabled)
ValidateServerCertificate (VSC)	1 (enabled)
WithHold Cursors (WH)	1 (enabled)
XML Describe Type (XDT)	-10

Accounting Info

Attribute: AccountingInfo (AI)

Description: Sets the CLIENT ACCTNG register on the server.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values: *string*

where *string* is an accounting ID.

Default: None

GUI Tab: Advanced tab on page 186

Add to Create Table

Attribute: AddStringToCreateTable (ASCT)

Description: A string that is automatically added to all Create Table statements. This option is for users who need to add an In Database clause to Create Table statements.

Valid Values: *string*

where *string* is valid syntax for the In Database clause of a Create Table statement.

Default: None

GUI Tab: Advanced tab on page 186

Alternate ID

Attribute: AlternateID (AID)

Description: The name of the default schema that is used to qualify unqualified database objects in dynamically prepared SQL statements. If the attempt to change the current schema fails, the connection fails and you receive the message `Invalid value for Alternate ID`. DB2 permissions must be set to SYSADM. (This option is not valid for DB2 V5R1 for iSeries.)

Valid Values: *string*

where *string* is a valid DB2 schema name.

Default: None

GUI Tab: Advanced tab on page 186

Alternate Servers

Attribute AlternateServers (ASVR)

Description A list of alternate database servers to which the driver will try 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 `(IPAddress=`*`ipvalue`*`:TcpPort=`*`portvalue`*`:{Database | Location}=`*`databasevalue`*`[, . . .])`

You must specify the IP address, port number, and database name (Linux/UNIX/Windows) or location (z/OS and iSeries) of each alternate server.

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

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

```
AlternateServers=(IpAddress=123.456.78.90:
TcpPort=5177:Database=DB2DAT, IpAddress=
223.456.78.90:TcpPort=5178:Database=DB2DAT3)
```

or

```
AlternateServers=(IpAddress=123.456.78.90:
TcpPort=5177:Location=DB2DAT, IpAddress=
223.456.78.90:TcpPort=5178:Location=DB2DAT3)
```

Default None

GUI Tab Failover tab on page 192

Application Name

Attribute ApplicationName (AN)

Description Sets the CLIENT APPLNAME register on the server. On DB2 for Linux/UNIX/Windows V9.1 and higher, this option also sets the APPL_NAME column of the SYSIBMADM.APPLICATIONS table.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values *string*

where *string* is an application name.

Default None

GUI Tab Advanced tab on page 186

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Advanced tab on page 186

Authentication Method

Attribute AuthenticationMethod (AM)

Description Specifies the method the driver uses to authenticate the user to the server when a connection is established. If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Valid Values 0 | 1 | 2 | 3 | 4

If set to 0 (No Encryption), the driver sends the user ID and password in clear text to the server for authentication.

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

If set to 2 (Encrypt UID and Password), the driver sends an encrypted user ID and 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), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

Default 0 (No Encryption)

GUI Tab Security tab on page 188

Batch Size

Attribute: BulkLoadBatchSize (BLBS)

Description: The number of rows at a time that the driver sends to the database 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.

If set to 0, no rows are sent.

If set to *x*, the specified number of rows are sent.

Default: 1024

GUI Tab: Bulk tab on page 196

Bulk Binary Threshold

Attribute: BulkBinaryThreshold (BBT)

Description: 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.

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 on page 196

Bulk Character Threshold

Attribute BulkCharacterThreshold (BCT)

Description 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.

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 on page 196

Catalog Schema

Attribute CatalogSchema (CS)

Description The DB2 schema to use for Catalog functions.

Valid Values *schema_name*

where *schema_name* is the name of a valid DB2 schema. If you do not specify a value for this attribute, the driver uses SYSIBM when connected to z/OS, QSYS2 when connected to iSeries, and SYSCAT when connected to Linux/UNIX/Windows.

Default None

GUI Tab Advanced tab on page 186

Character Set for CCSID 65535

Attribute CharsetFor65535 (CF6)

Description The IANA code page to be used by the driver to convert character data stored as bit data in character columns (Char, Varchar, Longvarchar, Clob, Char for Bit Data, Varchar for Bit Data, Longvarchar for Bit Data) defined with CCSID 65535.

Valid Values 0 | *IANA_code_page*

where *IANA_code_page* is a valid IANA code page. Refer to "IBM to IANA Code Page Values" on page 21 in Chapter 1 of the *DataDirect Connect Series for ODBC Reference* for a list of the most commonly used IBM code pages and their IANA code page equivalents.

If unspecified or set to 0, the driver returns these columns as binary columns (SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY) and does no conversion of the data.

If an IANA code page is specified, the driver returns these columns as character columns in the character set specified. The driver does no conversion of data supplied in bound parameters.

Default 0

GUI Tab Advanced tab on page 186

Client Host Name

Attribute ClientHostName (CHN)

Description Sets the CLIENT WRKSTNNAME register on the server.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values *client_name*

where *client_name* is the name of the client workstation.

Default None

GUI Tab Advanced tab on page 186

Client User

Attribute ClientUser (CU)

Description Sets the CLIENT USERID register on the server.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values -1 | *client_userid*

where *client_userid* is the current user ID.

The value -1 means that the driver uses the userid of the user that is currently logged onto the client.

Default None

GUI Tab Advanced tab on page 186

Collection

Attribute: Collection (COL)

Description: Valid only for DB2 for z/OS and iSeries. The current collection or library to use.

For DB2 for z/OS, this value is the user ID. If an attempt to change the current schema fails, the connection fails and you receive the message `Invalid value for Alternate ID`.

For DB2 for iSeries, this value is the name of the schema to be used as the default qualifier for unqualified object names. If you want to access a table outside of this schema, you must specify the schema name and the table name. For example:

```
SELECT * FROM Schema.Tablename
```

Also, if the Alternate ID option is set, it overrides this option.

NOTE: This option is mutually exclusive with the Database Name option.

Valid Values: *user_ID* (DB2 for z/OS) | *schema_name* (DB2 for iSeries)

where

user_ID is a valid user ID. DB2 permissions on the user ID must be set to SYSADM.

schema_name is the default schema to use for unqualified object names.

Default: None

GUI Tab: General tab on page 185

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 194

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 194

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 192

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

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

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

Default 3

GUI Tab Failover tab on page 192

Current Function Path

Attribute CurrentFunctionPath (CFP)

Description A comma-separated list of DB2 schema names used to resolve unqualified function names and data type references in dynamically prepared SQL statements. This value also is used to resolve unqualified stored procedure names specified in CALL statements.

Valid Values *schema_name*[,...]

where *schema_name* is the name of a valid DB2 schema.

Default None

GUI Tab Advanced tab on page 186

Data Source Name

Attribute: DataSourceName (DSN)

Description: 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 on page 185

Database Name

Attribute: Database (DB)

Description: The name of the database to which you want to connect.

Valid only for DB2 for Linux/UNIX/Windows.

This option is mutually exclusive with the Location Name and Collection options.

Valid Values: *database_name*

where *database_name* is the name of a valid database.

Default: None

GUI Tab: General tab on page 185

Default Isolation Level

Attribute: DefaultIsolationLevel (DIL)

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

The following table shows how ODBC isolation levels map to DB2 isolation levels.

ODBC	DB2
Read Uncommitted	Uncommitted Read
Read Committed	Cursor Stability
Repeatable Read	Read Stability
Serializable	Repeatable Read

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details about ODBC isolation levels.

Valid Values 0 | 1 | 2 | 3 | 4

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

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

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

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

table will be identical to the previous table; however, it can contain updates made by your application.

If set to 4 (NONE), your application can read modified records even if they have not been committed by another application. This level can only be set in the data source, not from the application. (This level is valid only on DB2 for iSeries, and is the only isolation level that works for collections that have journaling disabled.)

Default 1 (READ_COMMITTED)

GUI Tab Advanced tab on page 186

Description

Attribute Description (n/a)

Description 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 on page 185

Dynamic Sections

Attribute DynamicSections (DS)

Description The maximum number of prepared statements that the driver can have open at any time.

The driver only keeps the specified number of prepared statements open. If the driver detects that the number of dynamic sections available in the bound DB2 packages is less than

the number of dynamic sections requested in the connection string or data source, it generates the following message:

```
The current number of dynamic sections available for use is
different than the number of dynamic sections currently
specified in the connection string or data source.
```

Valid Values *x*

where *x* is a positive integer that specifies the number of prepared statements.

Default 200

GUI Tab Bindings tab on page 190

Enable Bulk Load

Attribute EnableBulkLoad (EBL)

Description Specifies the bulk load method.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses the database bulk load protocols. If the protocols cannot be used, the driver returns an error.

If set to 0 (Disabled), the driver uses standard parameter arrays.

Default 0 (Disabled)

GUI Tab Bulk tab on page 196

Encryption Method

Attribute EncryptionMethod (EM)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | 1 | 2

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

If set to 2 (Database Encryption), data is encrypted using the DB2 encryption protocol (supported only on DB2 for Linux/UNIX/Windows and DB2 for z/OS).

This option can only be set to 1 or 2 when Authentication Method is set to 0, 1, or 2.

Default 0 (No Encryption)

GUI Tab Security tab on page 188

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 192

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | 1 | 2

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 on page 192

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

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 on page 192

Grant Execute to [check box]

Attribute GrantExecute (GE)

Description Determines how EXECUTE privileges are granted on DB2 packages.

Valid Values 0 | 1

If set to 1 (Enabled), EXECUTE privileges are granted on DB2 packages that you are creating. By default, the schema to which privileges are granted is PUBLIC.

If set to 0 (Disabled), EXECUTE privileges are granted to the schema that created the DB2 packages.

Default 1 (Enabled)

GUI Tab Bindings tab on page 190

Grant Execute to [field]

Attribute GrantAuthid (GA)

Description Determines which DB2 schema is granted EXECUTE privileges for DB2 packages.

Valid Values *schema_name*

where *schema_name* is the name of a valid DB2 schema.

Default PUBLIC

GUI Tab Bindings tab on page 190

GSS Client Library

Attribute: GSSClientLibrary (GS)

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

The driver uses the standard path for loading the specified client library.

Valid Values: native | *client_library*

where *client_library* is a GSS client library installed on the client.

If set to *client_library*, the driver uses the specified GSS client library.

If set to native, the driver uses the GSS client shipped with the operating system.

Default: native

GUI Tab: Security tab on page 188

Host Name In Certificate

Attribute: HostNameInCertificate (HNIC)

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

Valid Values: *host_name* | #SERVERNAME#

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

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 Common Name parts.

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

Default None

GUI Tab Security tab on page 188

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 “Internationalization, Localization, and Unicode” in the *DataDirect Connect Series for ODBC Reference* for details.

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

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

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

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 186

Ip Address

Attribute IpAddress (IP)

Description Identifies the machine where catalog tables are stored.

Valid Values *host_name* | *IP_address*

where

host_name is the host name of the machine where catalog tables are stored. The driver must be able to find this name (with the correct address assignment) in the HOSTS file on the workstation or in a DNS server.

IP_address is the IP address of the machine where catalog tables are stored. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details about these formats.

Default

localhost

GUI Tab General tab on page 185

Key Password

Attribute KeyPassword (KP)

Description The password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values *key_password*

where *key_password* is the password of a key in the keystore.

Default None

GUI Tab Security tab on page 188

Keystore

Attribute Keystore (KS)

Description The directory of the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file.

Valid Values *keystore_directory*

where *keystore_directory* is the location of the keystore file.

Default None

GUI Tab Security tab on page 188

Keystore Password

Attribute KeystorePassword (KSP)

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

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

Valid Values *keystore_password*

where *keystore_password* is the password of the keystore file.

Default None

GUI Tab Security tab on page 188

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

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

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

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

Default 0

GUI Tab Pooling tab on page 194

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 194

Location Name

Attribute Location (LOC)

Description Valid only for DB2 for z/OS and iSeries. The name of the DB2 location that you want to access.

On DB2 for z/OS, your system administrator can determine the name of your DB2 location using the `DISPLAY DDF` command.

On DB2 for iSeries, your system administrator can determine the name of your DB2 location using the `WRKRDBDIRE` command. The name of the database that is listed as *LOCAL is the value to use.

This option is mutually exclusive with the Database Name option.

Valid Values *`location_name`*

where *`location_name`* is the name of a valid DB2 location.

Default None

GUI Tab General tab on page 185

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

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

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 186

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values: An integer from 1 to 65535

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

Default: 100

GUI Tab: Pooling tab on page 194

Min Pool Size

Attribute: MinPoolSize (MNPS)

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

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values: 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

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

Default: 0

GUI Tab: Pooling tab on page 194

Package Collection

Attribute PackageCollection (PC)

Description The name of the DB2 collection or location where the driver creates bind packages and, when required, searches for them.

Valid Values *collection_name*

where *collection_name* is a valid DB2 collection or location name.

Default NULLID

GUI Tab Bindings tab on page 190

Package Name Prefix

Attribute PackageNamePrefix (PNP)

Description A two-character prefix to be used for package names the driver uses for executing dynamic SQL. The default package name uses the following syntax:

DDOiVRMx

where:

DD is the two-character prefix.

i is one of the following characters:

- S—Serializable : DB2 RR
- R—Repeatable Read : DB2 RS
- C—Commited Read : DB2 CS
- U—Uncommited Read : DB2 UC
- N—Not committed: DB2 NC

VRM is the Version Release Modification, for example, you can specify 520 to represent version 5.2.0.

x is a one-character suffix that specifies:

- A—Cursor queries/updates
- B—Cursor queries/updates with hold
- C—Stored procedures (section 1 is for stored procedures that do not have parameters; section 2 is for procedures that do have parameters)

For example, the package name DDOC520A would represent a package using the Committed Read isolation level, at version 5.20, and using cursor queries/updates.

Valid Values *xx*

where *xx* is a two-character prefix.

Default DD

GUI Tab Advanced tab on page 186

Package Owner

Attribute PackageOwner (PO)

Description The AuthID assigned to the package.

Valid Values *authid*

where *authid* is a valid DB2 AuthID that has permissions to execute all the SQL in the package.

Default None

GUI Tab Bindings tab on page 190

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the

driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Program ID

Attribute ProgramID (PID)

Description Sets the Client Product Version/ID on the server. On DB2 for Linux/UNIX/Windows V9.1 and higher, this value is located in the CLIENT_PRDID column of the SYSIBMADM.APPLICATIONS table.

This connection option can affect performance. See "Performance Considerations" on page 250 for details.

Valid Values DDT*VVRRM*

where:

DDT identifies a DataDirect Connect driver.

VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version).

RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release).

M identifies a 1-character modification level (0-9 or A-Z).

Example DDT06000

Default None

GUI Tab Advanced tab on page 186

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.

Valid Values: -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Query timeout is supported on DB2 for Linux/UNIX/Windows 8.1 and higher and on DB2 for z/OS 8.1 and higher

Default: 0

GUI Tab: Advanced tab on page 186

Report Codepage Conversion Errors

Attribute: ReportCodepageConversionErrors (RCCE)

Description: 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

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 on page 186

Tcp Port

Attribute TcpPort (PORT)

Description The port number that is assigned to the DB2 DRDA listener process on the server host machine.

On DB2 for iSeries only, execute `NETSTAT` from a command line to determine the correct port number. Select option 3 to display a list of active ports on the iSeries machine. Find the entry for DRDA, and press F14 to toggle and display the port number. If DRDA is not currently listening, the command `CHGDDMTCPA AUTOSTART(*YES) PWDRQD(*YES)` starts the listener and ensures that it is active at IPL.

Valid Values *IP_address* | *service_name*

where:

IP_address is the port's IP address.

service_name is the port's service name. The driver must be able to find this name (with the correct port assignment) in the SERVICES file on the workstation.

Default 50000

GUI Tab General tab on page 185

Truststore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 188

Truststore Password

Attribute TruststorePassword (TSP)

Description The password that is used to access the truststore file when SSL is enabled (Encryption Method=1) and server authentication is

used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

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

Valid Values *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default None

GUI Tab Security tab on page 188

Use Current Schema for Catalog Functions

Attribute UseCurrentSchemaforCatalogFunctions (UCS)

Description Specifies whether results are restricted to the tables and views in the current schema if a catalog function call is made without specifying a schema or if the schema is specified as the wildcard character %. Restricting results to the tables and views in the current schema improves performance of catalog calls that do not specify a schema.

Valid Values 0 | 1

If set to 1 (Enabled), results of catalog function calls are restricted to the tables and views in the current schema.

If set to 0 (Disabled), results of catalog function calls are not restricted.

Default 0 (Disabled)

GUI Tab Advanced tab on page 186

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 on page 188

Validate Server Certificate

Attribute: ValidateServerCertificate (VSC)

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

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

Valid Values: 0 | 1

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate

option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

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

Default 1 (Enabled)

GUI Tab Security tab on page 188

With Hold Cursors

Attribute WithHold Cursors (WH)

Description Determines whether the cursor stays open on a commit.

Valid Values 0 | 1

If set to 1 (Enabled), cursor behavior is Preserve, which keeps cursors open after a commit or rollback (SQLGetInfo( ) returns SQL_CB_PRESERVE for SQL_COMMIT_CURSOR_BEHAVIOR).

If set to 0 (Disabled), cursor behavior is Delete, which closes all cursors open after a commit or rollback (SQLGetInfo( ) returns SQL_CB_DELETE).

Default 1 (Enabled)

GUI Tab Advanced tab on page 186

XML Describe Type

Attribute XML Describe Type (XDT)

Description The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See "Using the XML Data Type" on page 254 for further information about the XML data type.

Valid Values -4 | -10

If set to -4 (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to -10 (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

-10

GUI Tab Advanced tab on page 186

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: 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.

Enable Bulk Load (EnableBulkLoad): If your application performs bulk loading of data, you can improve performance by configuring the driver to use the database system's bulk load functionality instead of database array binding. The trade-off to consider for improved performance is that using the bulk load functionality can bypass data integrity constraints.

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

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

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.

Workload Manager: The Workload Manager (WLM) automatically adjusts server resources, such as CPU and memory, based on the service class associated with a DB2 workload. Therefore, an application's performance is tied to the DB2 workload to which it is assigned and, ultimately, to the service class associated with that workload. The DB2 Wire Protocol driver allows your application to set client information in the DB2 database that can be used by the WLM to classify work. If you know that your database environment uses WLM, coordinate with your database administrator to determine how setting the following options affects performance.

- **Accounting Info**: Sets the CLIENT ACCTNG register on the server.
- **Application Name**: Sets the CLIENT APPLNAME register on the server.
- **Client Host Name**: Sets the CLIENT WRKSTNNAME register on the server.
- **Client User**: Sets the CLIENT USERID register on the server.
- **Program ID**: Sets the Client Product Version/ID on the server.

IBM to IANA Code Page Values

Refer to "IBM to IANA Code Page Values" on page 21 in Chapter 1 of the *DataDirect Connect Series for ODBC Reference* for a list of the most commonly used IBM code pages and their IANA code page equivalents. The IANA values are valid for the CharsetFor65535 connection string attribute and the Character Set for the CCSID 65535 option.

Data Types

Table 5-4 shows how the DB2 data types map to the standard ODBC data types. "Unicode Support" on page 255 lists DB2 to Unicode data type mappings.

Table 5-4. DB2 Data Types

DB2	ODBC
Bigint [1]	SQL_BIGINT
Blob [2]	SQL_LONGVARBINARY
Char	SQL_CHAR
Char() for Bit Data	SQL_BINARY
Clob [3]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decfloat [4]	SQL_DOUBLE
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY

Table 5-4. DB2 Data Types (cont.)

DB2	ODBC
Numeric	SQL_NUMERIC
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR
Varchar() for Bit Data	SQL_VARBINARY
XML [5]	SQL_LONGVARCHAR

1. Supported on DB2 v8.x and higher for Linux/UNIX/Windows, DB2 v9 and higher for z/OS, and DB2 V5R2 and higher for iSeries.
2. Supported on DB2 v8.x and higher for Linux/UNIX/Windows; DB2 for z/OS; and DB2 V5R2 and higher for iSeries.
3. On DB2 for Linux/UNIX/Windows versions previous to v8.1 and DB2 V5R2 for iSeries, only the first 32 KB of the Clob data are returned when fetching, and only 32 KB can be inserted and updated.
4. Supported only on DB2 V9 and higher for Linux/UNIX/Windows, DB2 v9 and higher for z/OS, and DB2 V6R1 for iSeries.
5. Supported only on DB2 V9.1 and higher for Linux/UNIX/Windows.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Using the XML Data Type

By default, DB2 returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the attribute XML Describe Type to

SQL_LONGVARBINARY (-10) and bind the data as SQL_C_BINARY.

Unicode Support

The DB2 Wire Protocol driver supports Unicode data types if the database was created with a multi-byte character set.

The DB2 Wire Protocol driver maps the DB2 data types to Unicode data types as shown in the following table:

DB2 Data Type	Mapped to. . .
Dbclob [1]	SQL_WLONGVARCHAR
Graphic	SQL_WCHAR
Long Vargraphic	SQL_WLONGVARCHAR
Vargraphic	SQL_WVARCHAR

1. Supported on DB2 v8.x and higher for Linux/UNIX/Windows; DB2 for z/OS; and DB2 V5R2 and higher for iSeries.

Unexpected Characters

Users are sometimes surprised when they insert a character into a database, only to have a different character displayed when they fetch it from the database. There are many reasons this can happen, but it most often involves code page issues, not driver errors.

Client and server machines in a database system each use code pages, which can be identified by a name or a number, such as Shift_JIS (Japanese) or cp1252 (Windows English). A code page is a mapping that associates a sequence of bits, called a code point, with a specific character. Code pages include the characters and

symbols of one or more languages. Regardless of geographical location, a machine can be configured to use a specific code page. Most of the time, a client and database server would use similar, if not identical, code pages. For example, a client and server might use two different Japanese code pages, such as Shift_JIS and EUC_JP, but they would still share many Japanese characters in common. These characters might, however, be represented by different code points in each code page. This introduces the need to convert between code pages to maintain data integrity. In some cases, no one-to-one character correspondence exists between the two code points. This causes a substitution character to be used, which can result in displaying an unexpected character on a fetch.

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage attribute (see "IANAAppCodePage" on page 233). 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 client. 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 client machine is running the Japanese code page EUC_JP.
- The DB2 server is running the Japanese code page Shift_JIS.
- When you insert the EUC_JP code point 0xA1BD and then fetch it back, you do not see the character you expected. In

fact, what displays on the client may not be a recognizable character.

This substitution occurs because the code points do not correspond in the two code pages. EUC_JP code point 0xA1BD is converted to UTF-16 code point 0x2014. Code point 0x2014 does not map to anything in Shift_JIS, resulting in the Shift_JIS substitution code point, 0x3F, being sent to, and stored in, the database. When this character is retrieved, depending on the client display, it may not display as a recognizable character.

This is not a driver error. It occurs because the code points map differently and because some characters do not exist in a code page. The best way to avoid these problems is to use the same code page on both the client and server machines.

XQuery Expressions

The DB2 Wire Protocol driver supports execution of XQuery expressions in DB2 V9.1 and higher for Linux/UNIX/Windows. IBM provides a tutorial on this topic at the following URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

Click the Tutorial link in the right-hand pane of the browser window.

Stored Procedure Support

The DB2 Wire Protocol driver supports DB2 Remote Procedure Calls (RPCs) as follows:

- Multiple result sets are returned.
- RPCs must take an argument list. The driver does not support RPCs that use a SQL descriptor area (SQLDA) data structure to specify the arguments.
- Literals are supported as stored procedure parameters.

Using DataDirect Bulk Load on DB2

DataDirect Bulk Load offers a simple, consistent way to do bulk load operations. See "Using DataDirect Bulk Load" on page 115 for details.

The DB2 driver has no special requirements for bulk load operations.

Using Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

DB2 supports isolation level 0 (read uncommitted), isolation level 1 (read committed), isolation level 2 (repeatable read), isolation level 3 (serializable), and, on DB2 for iSeries only, isolation level 4 (none).

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The following X/Open functions are supported:

- SQLProcedures
- SQLProcedureColumns

The driver supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the DB2 Wire Protocol driver.

Number of Connections and Statements Supported

The DB2 database system supports multiple connections and multiple statements per connection.

Using Arrays of Parameters

DB2 for Linux/UNIX/Windows natively supports parameter arrays, and the DB2 Wire Protocol driver, in turn, supports them when connected to these DB2 databases. When designing an application for performance, using native parameter arrays for bulk inserts or updates, for example, can improve performance. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

If the database does not support parameter arrays, the DB2 Wire Protocol driver emulates them so that you can design your applications to use arrays of parameters and take advantage of the performance improvements where applicable. The driver emulates parameter arrays by sending individual rows to the database.

6 The Greenplum Wire Protocol Driver

The DataDirect Connect Series *for* ODBC Greenplum Wire Protocol driver (the Greenplum Wire Protocol driver) is available in a 32-bit version both 32- and 64-bit versions and supports the following Greenplum database server:

Greenplum version 3, 3.2, and 3.3

The Greenplum Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the Windows, UNIX, and Linux environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Greenplum Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Connection Pooling

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 274 and Table 6-1 on page 277 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 6-1 on page 277 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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

To configure a Greenplum data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

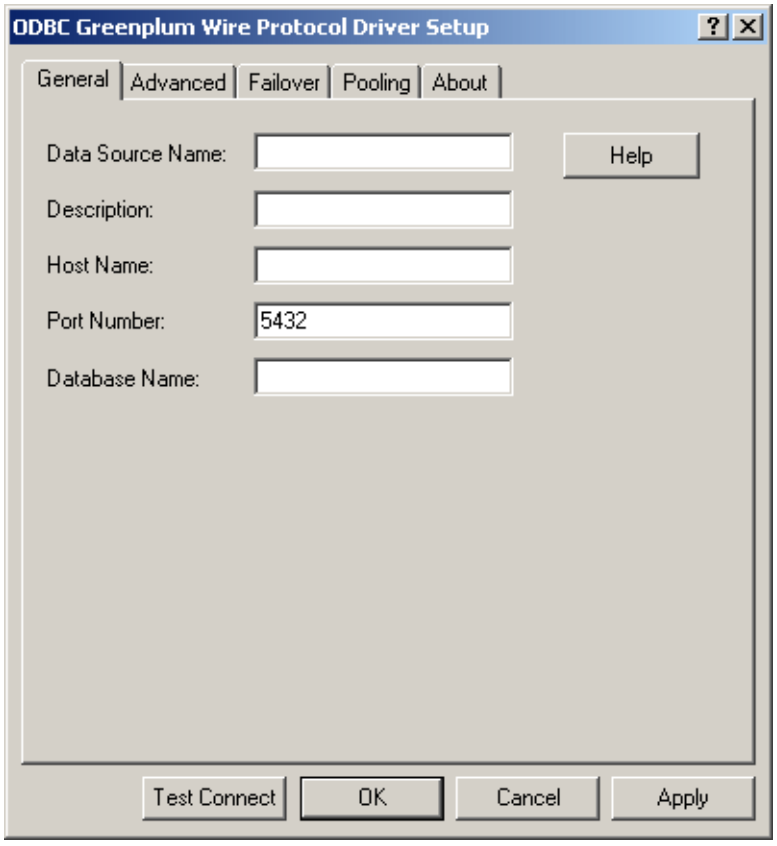
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

The General tab of the Setup dialog box appears.

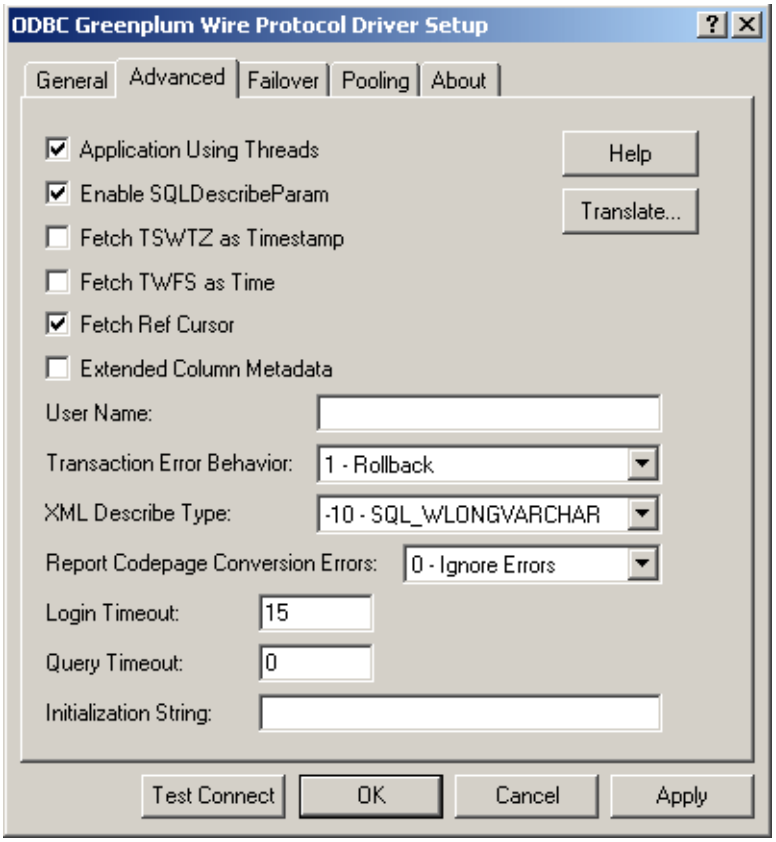


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 282)	None
Description (see page 283)	None
Host Name (see page 289)	None
Port Number (see page 295)	5432
Database Name (see page 282)	None

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



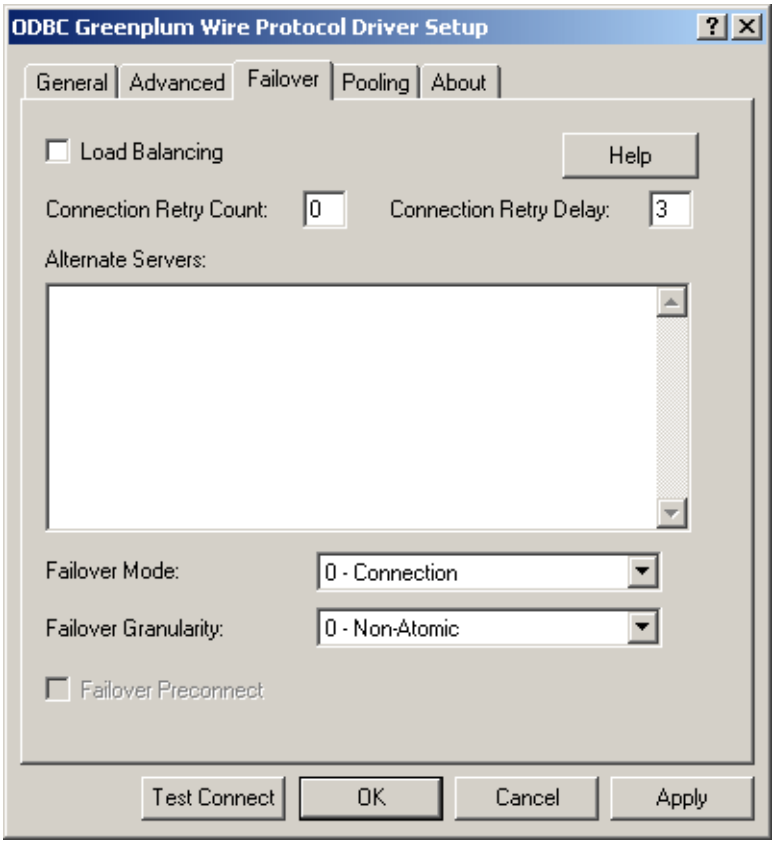
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Using Threads (see page 279)	Enabled
Enable SQLDescribeParam (see page 283)	Disabled
Fetch TSWTZ as Timestamp (see page 288)	Disabled
Fetch TWFS as Time (see page 288)	Disabled
Fetch RefCursors (see page 287)	Enabled
Extended Column Metadata (see page 284)	Disabled
User Name (see page 298)	None
Transaction Error Behavior (see page 297)	1 - Rollback
XML Describe Type (see page 298)	-10 SQL_WLONGVARCHAR
Report Codepage Conversion Errors (see page 296)	0 - Ignore Errors
Login Timeout (see page 293)	15
Query Timeout (see page 295)	0
Initialization String (see page 291)	None
IANAAppCodePage (see page 290) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

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



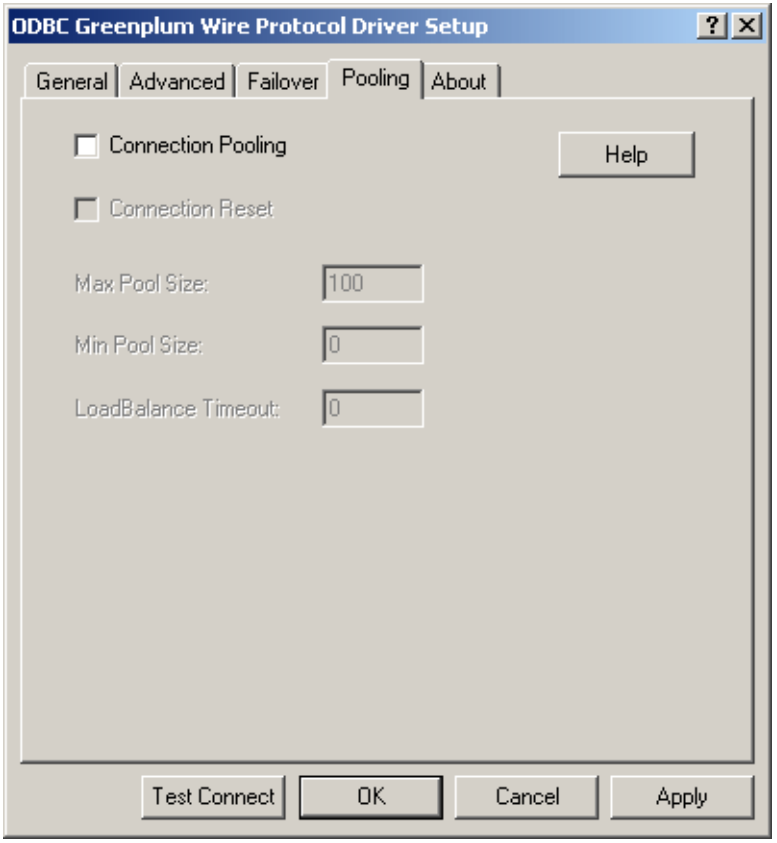
See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 292)	Disabled
Connection Retry Count (see page 281)	0
Connection Retry Delay (see page 281)	3

Connection Options: Failover *(cont.)*	Default
Alternate Servers (see page 278)	None
Failover Mode (see page 286)	0 - Connection
Failover Granularity (see page 285)	0 - Non-Atomic
Failover Preconnect (see page 286)	Disabled

6 Optionally, click. the **Pooling** tab to specify pooling data source settings.



See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 279)	Disabled
Connection Reset (see page 280)	Disabled
Max Pool Size (see page 293)	100
Min Pool Size (see page 294)	0
Load Balance Timeout (see page 291)	0

7 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 275 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

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

8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 6-1 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Greenplum Wire Protocol is:

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

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

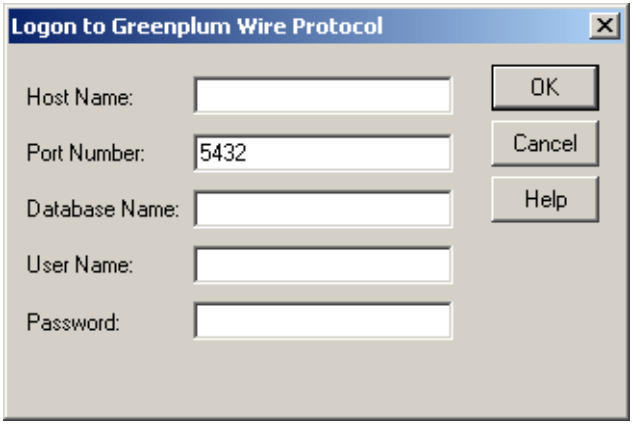
```
FILEDSN=GreenplumWP.dsn;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect Greenplum Wire Protocol;
HOST=GreenplumServer;PORT=5432;UID=JOHN;PWD=XYZZY;DB=
Gplumdb1
```

Using a Logon Dialog Box

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



In this dialog box, provide the following information:

1. In the Host Name field, type either the name or the IP address of the server to which you want to connect. The IP address must be in IPv4 format.
2. In the Port Number field, type the number of your Greenplum listener. Check with your database administrator for the correct number.
3. In the Database Name field, type the name of the database to which you want to connect.
4. If required, type your Greenplum user name.
5. If required, type your Greenplum password.
6. Click **OK** to log on to the Greenplum database installed on the server you specified and to update the values in the Registry.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

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

Table 6-1 lists the connection string attributes supported by the Greenplum Wire Protocol driver.

Table 6-1. Greenplum Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
EnableDescribeParam (EDP)	0 (Disabled)
ExtendedColumnMetaData (ECMD)	0 (Disabled)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchRefCursors= (FRC)	1 (Enabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
HostName (HOST)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100

Table 6-1. Greenplum Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
MinPoolSize (MNPS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
PortNumber (PORT)	5432
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
TransactionErrorBehavior (TEB)	1 (Rollback Transaction)
XML Describe Type (XDT)	-10

Alternate Servers

Attribute AlternateServers (ASVR)

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

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

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

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

```
AlternateServers=(HostName=GreenplumServer:
PortNumber=5431:Database=Pgredb1,
HostName=255.201.11.24:PortNumber=5432:Database=Pgredb2)
```

Default None

GUI Tab Failover tab on page 271

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Advanced tab on page 269

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 272

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 272

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 271

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

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

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

Default 3

GUI Tab Failover tab on page 271

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 269

Database Name

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 269

Description

Attribute Description (n/a)

Description 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 on page 269

Enable SQLDescribeParam

Attribute EnableDescribeParam (EDP)

Description Determines whether SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

Valid Values 0 | 1

If set to 1 (Enabled), SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

If set to 0 (Disabled), the driver does not support SQLDescribeParam and returns the message: `Driver does not support this function`.

Default 0 (Disabled)

GUI Tab Advanced tab on page 269

Extended Column Metadata

Attribute ExtendedColumnMetaData (ECMD)

Description Determines how the driver returns column metadata when using SQLDescribCol and SQLColAttribute.

Valid Values 0 | 1

If set to 1 (Enabled), SQLDescribCol returns the actual values for Data Type, Column Size, Decimal Digits, and Nullable. SQLColAttribute returns the actual values for:

- SQL_DESC_CATALOG_NAME
- SQL_DESC_TABLE_NAME
- SQL_DESC_BASE_COLUMN_NAME
- SQL_DESC_LOCAL_TYPE_NAME
- SQL_DESC_NULLABLE
- SQL_DESC_AUTO_UNIQUE_VALUE

If set to 0 (Disabled), SQLDescribCol returns the Data Type, Column Size, and Decimal Digits for the column. The value SQL_NULLABLE_UNKNOWN is returned for Nullable. SQLColAttribute returns the following attribute values:

- SQL_DESC_CATALOG_NAME: empty string
- SQL_DESC_TABLE_NAME: empty string
- SQL_DESC_BASE_COLUMN_NAME: empty string
- SQL_DESC_LOCAL_TYPE_NAME: empty string
- SQL_DESC_NULLABLE: SQL_NULLABLE_UNKNOWN
- SQL_DESC_AUTO_UNIQUE_VALUE: SQL_FALSE

Default 0 (Disabled)

GUI Tab Advanced tab on page 269

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 271

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

Valid Values 0 | 1 | 2

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 on page 271

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

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 on page 271

Fetch RefCursors

Attribute FetchRefCursors= (FRC)

Description Determines whether the driver returns refcursors from stored procedures as results sets.

Valid Values 0 | 1

If set to 1 (Enabled), the driver returns refcursors from stored procedures as result sets. The driver fetches all the data from the refcursor and then closes the refcursor. If a stored procedure returns multiple refcursors, the driver generates multiple result sets, one for each refcursor returned.

If set to 0 (Disabled), the driver returns the cursor name for refcursors. The application must fetch the actual data from the refcursor using the cursor name and must close the cursor before additional processing can be done on the statement. The application must close the cursor regardless of whether it actually fetches data from the cursor.

Default 1 (Enabled)

GUI Tab Advanced tab on page 269

Fetch TSWTZ as Timestamp

Attribute FetchTSWTZasTimestamp (FTSWTZAT)

Description Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values 0 | 1

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 timezone 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 timezone information in the fetched value.

Default 0 (Disabled)

GUI Tab Advanced tab on page 269

Fetch TWFS as Time

Attribute FetchTWFSasTime (FTWFSAT)

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

Valid Values 0 | 1

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

If set to 0 (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP. The

fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp

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

Default 0 (Disabled)

GUI Tab Advanced tab on page 269

Host Name

Attribute HostName (HOST)

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

Valid Values *server_name* | *IP_address*

where

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

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

The IP address must be in IPv4 format.

Default None

GUI Tab General tab on page 269

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

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

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

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

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 269

Initialization String

Attribute InitializationString (IS)

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

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

Valid Values *SQL_command*

where *SQL_command* is a valid SQL command that is supported by the database.

Example To set the date format on every connection, specify:

```
Set DateStyle='ISO, MDY'
```

Default None

GUI Tab Advanced tab on page 269

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

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

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

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

Default 0

GUI Tab Pooling tab on page 272

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 271

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

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

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 269

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values An integer from 1 to 65535

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

Default 100

GUI Tab Pooling tab on page 272

Min Pool Size

Attribute MinPoolSize (MNPS)

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

This connection option can affect performance. See "Performance Considerations" on page 299 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

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

Default 0

GUI Tab Pooling tab on page 272

Password

Attribute: Password (PWD)

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

Valid Values: *pwd*

where *pwd* is a valid password.

Default: None

GUI Tab: n/a

Port Number

Attribute: PortNumber (PORT)

Description: The port number of the server listener.

Valid Values: *port_name*

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

Default: 5432

GUI Tab: General tab on page 269

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI Tab Advanced tab on page 269

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 269

Transaction Error Behavior

Attribute TransactionErrorBehavior (TEB)

Description Determines how the driver handles errors that occur within a transaction. When an error occurs in a transaction, the Greenplum server does not allow any operations on the connection except for rolling back the transaction.

Valid Values 0 | 1

If set to 0 (None), the driver does not roll back the transaction when an error occurs. The application must handle the error and roll back the transaction. Any operation on the statement other than a rollback results in an error.

If set to 1 (Rollback Transaction), the driver rolls back the transaction when an error occurs. In addition to the original error message, the driver posts an error message indicating that the transaction has been rolled back.

Default 1 (Rollback Transaction)

GUI Tab Advanced tab on page 269

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 Advanced tab on page 269

XML Describe Type

Attribute XML Describe Type (XDT)

Description The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See "Using the XML Data Type" on page 302 for further information about the XML data type.

Valid Values -4 | -10

If set to -4 (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to -10 (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

-10

GUI Tab Advanced tab on page 269

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: A connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size, if one has been specified. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

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

Data Types

Table 6-2 shows how the Greenplum data types are mapped to the standard ODBC data types. "Using the XML Data Type" on page 302 describes Greenplum to Unicode data type mappings.

Table 6-2. Greenplum Data Types

Greenplum	ODBC
Bigint	SQL_BIGINT
Bigserial	SQL_BIGINT
Bit [1]	SQL_BIT

***Table 6-2. Greenplum Data Types** (cont.)*

Greenplum	ODBC
Bit varying	SQL_VARBINARY
Boolean	SQL_BIT
Bytea	SQL_VARBINARY
Character	SQL_CHAR
Character varying	SQL_VARCHAR
Date	SQL_TYPE_DATE
Double Precision	SQL_DOUBLE
Integer	SQL_INTEGER
Name	SQL_VARCHAR
Numeric [2]	SQL_NUMERIC
Real	SQL_REAL
Serial	SQL_INTEGER
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR
Time [3]	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Timestamp with timezone [4]	SQL_VARCHAR
Xml	SQL_LONGVARCHAR

1. Bit maps to SQL_BIT when the length for the bit is 1. If the length is greater than 1, the driver maps the column to SQL_BINARY.
2. Numeric maps to SQL_NUMERIC if the precision of the Numeric is less than or equal to 38. If the precision is greater than 38, the driver maps the column to SQL_VARCHAR.
3. Time mapping changes based on the setting of the Fetch TWFS as Time option.
4. Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.

See "Retrieving Data Type Information" on page 77 for more information about data types.

Using the XML Data Type

By default, Greenplum returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the option XML Describe Type to SQL_LONGVARBINARY (-4) and bind the data as SQL_C_BINARY.

Unicode Support

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

Stored Procedure Results

Greenplum provides functionality to create user-defined functions. Greenplum does not make a distinction between user-defined functions and stored procedures. To Greenplum, everything is a user-defined function. Greenplum does not define a call mechanism for invoking a user-defined function. User-defined functions must be invoked via a SQL statement. For example, given a function defined as:

```
CREATE table foo (intcol int, varcharcol varchar(123))
CREATE or REPLACE FUNCTION insertFoo
(IN idVal int, IN nameVal varchar) RETURNS void
```

```
  AS $$
    insert into foo values ($1, $2);
  $$
  LANGUAGE SQL;
```

Must be invoked natively as:

```
SELECT * FROM insertFoo(100, 'Mark')
```

Even though the function does not return a value or results.

The Greenplum Wire Protocol driver supports invoking user-defined functions using the ODBC call Escape. The previously described function can be invoked using:

```
{call insertFoo(100, 'Mark')}
```

Greenplum functions return data from functions as a result set. If multiple output parameters are specified, the values for the output parameters are returned as columns in the result set. For example, the function defined as:

```
CREATE or REPLACE FUNCTION addValues(in v1 int, in v2 int)
  RETURNS int
  AS $$
    SELECT $1 + $2;
  $$
  LANGUAGE SQL;
```

returns a result set with a single column of type SQL_INTEGER, whereas the function defined as:

```
CREATE or REPLACE FUNCTION selectFooRow2
  (IN idVal int, OUT id int, OUT name varchar)
  AS $$
    select id, name from foo where id = $1;
  $$
  LANGUAGE SQL
```

returns a result set that contains two columns, a SQL_INTEGER id column and a SQL_VARCHAR name column.

In addition, when calling Greenplum functions that contain output parameters, the native syntax requires that the output parameter values be omitted from the function call. This, in addition to output parameter values being returned as a result set, makes the Greenplum behavior of calling functions different from most other databases.

The Greenplum Wire Protocol driver provides a mechanism that makes the invoking of functions more consistent with how other databases behave. In particular, the Greenplum Wire Protocol driver allows parameter markers for output parameters to be specified in the function argument list when the Escape call is used. The driver allows buffers to be bound to these output parameters. When the function is executed, the output parameters are removed from the argument list sent to the server. The driver extracts the output parameter values from the result set returned by the server and updates the bound output parameter buffers with those values. For example, the function `selectFooRow2` described previously can be invoked as:

```
sql = L"{call selectFooRow2(?, ?, ?)}";
retVal = SQLPrepare(hPrepStmt, sql, SQL_NTS);
retVal = SQLBindParameter(
  hPrepStmt, 1, SQL_PARAM_INPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf, 0, &idInd);
retVal = SQLBindParameter(
  hPrepStmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf2, 4, &idInd2);
retVal = SQLBindParameter(
  hPrepStmt, 3, SQL_PARAM_INPUT, SQL_C_WCHAR,
  SQL_VARCHAR, 30, 0, &nameBuf, 123, &nameInd);
retVal = SQLExecute(hPrepStmt);
```

The values of the id and name output parameters are returned in the `idBuf2` and `nameBuf` buffers.

If output parameters are bound to a function call, the driver returns the output parameters in the bound buffers. An error is returned if the number of output parameters bound when the function is executed is less than the number of output

parameters defined in the function. If no output parameters are bound to a function call, the driver returns the output parameters as a result set.

Greenplum can also return results from a function as a refcursor. There can be, at most, one refcursor per result; however, a function can return multiple results where each result is a refcursor. A connection option defines how the driver handles refcursors. See "Fetch RefCursors" on page 287 for details about this option.

Configuring Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

Greenplum supports isolation level 0 (read uncommitted), level 1 (read committed), 2 (Repeatable read), and level 3 (serializable). Greenplum supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The driver supports the core SQL grammar.

The Greenplum Wire Protocol driver also supports the following functions:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLTablePrivileges

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Greenplum Wire Protocol driver.

Number of Connections and Statements Supported

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

Using Arrays of Parameters

Greenplum supports returning a set of output parameters or return values, but no ODBC standard method exists for returning arrays of output parameters or return values. If the call Escape is used to invoke a function that returns a set of output parameters and buffers are bound for those output parameters, the Greenplum Wire Protocol driver places the first set of output parameters in the bound buffers. If no output parameters are bound for functions that return a set of results or output parameters, the driver returns a result set with a row for each set of output parameters.

7 The Informix Wire Protocol Driver

The DataDirect Connect Series *for* ODBC Informix Wire Protocol driver (the Informix Wire Protocol driver) is available in both 32- and 64-bit versions and supports multiple connections to the following Informix database servers:

Informix Dynamic Server 9.2*x*, 9.3*x,* 9.4*x*, 10*x*, and 11*x*.

The Informix Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the Informix Wire Protocol driver.

NOTE: The Informix Wire Protocol driver does not require any Informix client software. DataDirect also provides an Informix client-based driver that can access earlier versions of Informix databases. See "The Informix Driver" on page 899 for details.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 317 and Table 7-1 on page 320 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 7-1 on page 320 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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

To configure an Informix data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

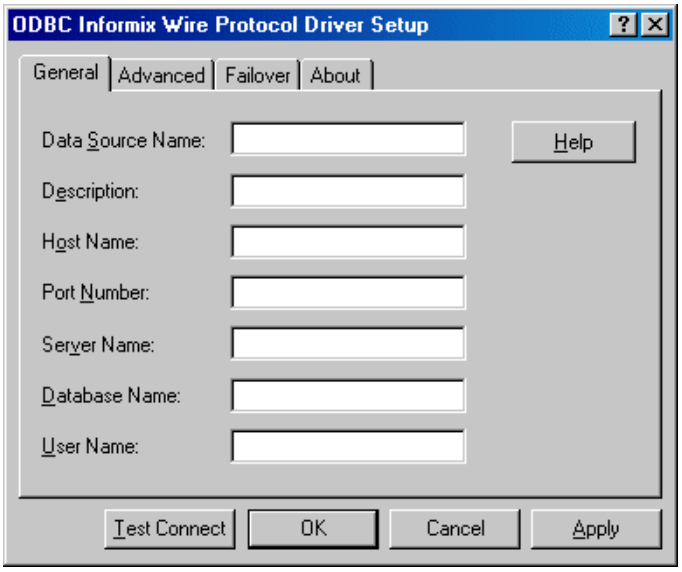
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

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

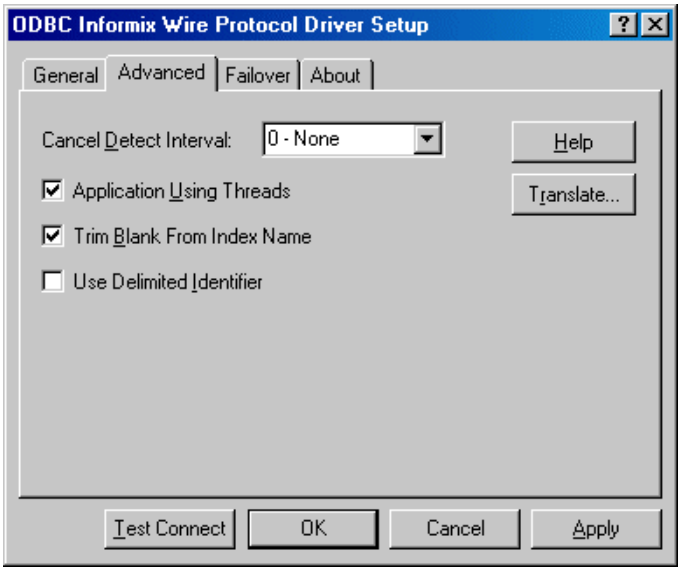


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 325)	None
Description (see page 326)	None
Host Name (see page 326)	None
Port Number (see page 329)	None
Server Name (see page 329)	None
Database Name (see page 325)	None
User Name (see page 330)	None

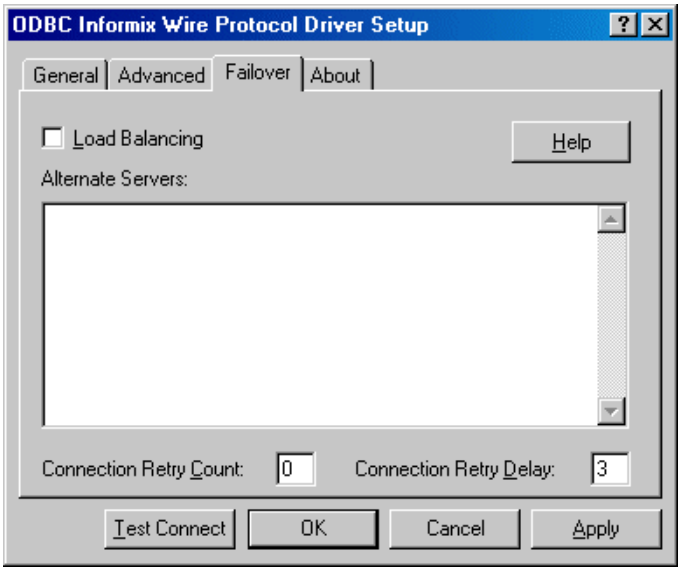
4 Optionally, click the **Advanced** tab to specify additional data source settings.



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

Connection Options: Advanced	Default
Cancel Detect Interval (see page 323)	0 - None
Application Using Threads (see page 322)	Enabled
Trim Blank From Index Name (see page 329)	Enabled
Use Delimited Identifier (see page 330)	Disabled
IANAAppCodePage (see page 327) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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



See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 327)	Disabled
Alternate Servers (see page 321)	None
Connection Retry Count (see page 323)	0
Connection Retry Delay (see page 324)	3

6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 319 for details). Note that the information you

enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

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

7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 7-1 on page 320 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 Informix is:

```
DSN=INFORMIX TABLES;DB=PAYROLL
```

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

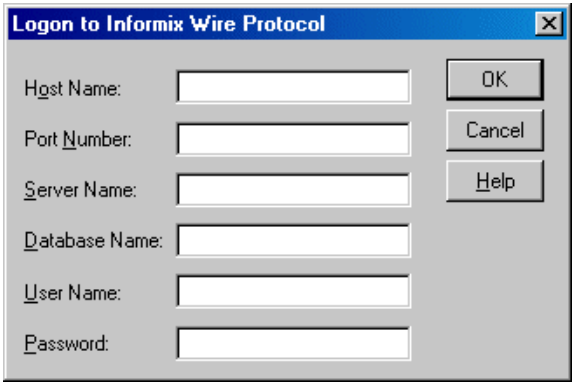
```
FILEDSN=Informix.dsn;DB=DBPAYROLL
```

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

```
DRIVER=DataDirect Informix Wire Protocol;
HOST=INF2;PORT=4321;SRVR=ACCT;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box

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



In this dialog box, provide the following information:

1. In the Host Name field, type the name or IP address of the host machine on which the Informix server resides.

   The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details concerning these formats.

2. In the Port Number field, type the port number of the server listener.

3. In the Server Name field, type the name of the Informix server.

4. In the Database Name field, type the name of the database to which you want to connect.

5. If required, type your user name as specified on the Informix server.

6 If required, type your password.

7 Click **OK** to complete the logon and to update these values in the Registry.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

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

Table 7-1 lists the connection string attributes supported by the Informix Wire Protocol driver.

Table 7-1. Informix Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
CancelDetectInterval (CDI)	0 (None)

Table 7-1. Informix Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
HostName (HOST)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
LoadBalancing (LB)	0 (Disabled)
LogonID (UID)	None
Password (PWD)	None
PortNumber (PORT)	None
ServerName (SRVR)	None
TrimBlankFromIndexName (TBFIN)	1 (Enabled)
UseDelimitedIdentifier (UDI)	0 (Disabled)

Alternate Servers

Attribute AlternateServers (ASVR)

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

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

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

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

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

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

Default None

GUI Tab Failover tab on page 316

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 331 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Advanced tab on page 315

Cancel Detect Interval

Attribute CancelDetectInterval (CDI)

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

This connection option can affect performance. See "Performance Considerations" on page 331 for details.

Valid Values 0 | *x*

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

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

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

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

Default 0 (None)

GUI Tab Advanced tab on page 315

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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

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

Default 0

GUI Tab See Failover tab on page 316

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

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

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

Default 3

GUI Tab See Failover tab on page 316

Database Name

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 314

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 314

Description

Attribute: Description (n/a)

Description: 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 on page 314

Host Name

Attribute: HostName (HOST)

Description: 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" on page 70 for details about these formats.

Default

None

GUI Tab: General tab on page 314

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled and/or if your database character set is not Unicode (refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details). The value you specify must match the database character encoding and the system locale.

The Driver Manager checks for the value of IANAAppCodePage in the following order:

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 315

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 316

Password

Attribute Password (PWD)

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

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

Valid Values *port_name*

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

Default None

GUI Tab General tab on page 314

Server Name

Attribute ServerName (SRVR)

Description The name of the Informix server.

Valid Values *server_name*

where *server_name* is a name that uniquely identifies the Informix server.

Default None

GUI Tab General tab on page 314

Trim Blank From Index Name

Attribute TrimBlankFromIndexName (TBFIN)

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

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

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

Default 1 (Enabled)

GUI Tab Advanced tab on page 315

Use Delimited Identifier

Attribute UseDelimitedIdentifier (UDI)

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

Valid Values 0 | 1

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

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

Default 0 (Disabled)

GUI Tab Advanced tab on page 315

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 General tab on page 314

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

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

Data Types

Table 7-2 shows how the Informix data types map to the standard ODBC data types.

Table 7-2. Informix Data Types

Informix	ODBC
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
BYTE	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB	SQL_LONGVARCHAR
DATE	SQL_TYPE_DATE
DATETIME YEAR TO FRACTION(f) [1]	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO SECOND	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO DAY	SQL_TYPE_DATE
DATETIME HOUR TO SECOND	SQL_TYPE_TIME
DATETIME HOUR TO FRACTION(f) [1]	SQL_TYPE_TIME
DECIMAL	SQL_DECIMAL
FLOAT	SQL_DOUBLE
INT8	SQL_BIGINT
INTEGER	SQL_INTEGER
INTERVAL YEAR(p) TO YEAR	SQL_INTERVAL_YEAR
INTERVAL YEAR(p) TO MONTH	SQL_INTERVAL_YEAR_TO_MONTH
INTERVAL MONTH(p) TO MONTH	SQL_INTERVAL_MONTH
INTERVAL DAY(p) TO DAY	SQL_INTERVAL_DAY
INTERVAL DAY(p) TO HOUR	SQL_INTERVAL_DAY_TO_HOUR
INTERVAL DAY(p) TO MINUTE	SQL_INTERVAL_DAY_TO_MINUTE
INTERVAL DAY(p) TO SECOND	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL DAY(p) TO FRACTION(f) [1]	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL HOUR(p) TO HOUR	SQL_INTERVAL_HOUR

Table 7-2. Informix Data Types (cont.)

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

1. (f) can have a value of 1, 2, 3, 4, or 5. The precision is type-dependent and the scale is 5.
2. Supported only on Informix 9.4 and higher servers.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

MTS Support

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

NOTE: The DataDirect Connect *for* ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 *for* ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

Informix supports isolation levels 0 (read uncommitted), 1 (read committed), and 3 (serializable). Informix supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The Informix Wire Protocol driver supports the following functions:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Informix Wire Protocol driver.

Number of Connections and Statements Supported

The Informix Wire Protocol driver supports multiple connections and multiple statements per connection to the Informix database system.

8 The MySQL Wire Protocol Driver

The DataDirect Connect Series *for* ODBC MySQL Wire Protocol driver (the MySQL Wire Protocol driver) is available in both 32- and 64-bit versions and supports multiple connections to the following server and storage engines:

- MySQL 5.1 server
- MySQL 5.0.*x* server
- Storage engines
  - InnoDB – Transactional
  - MyISAM – Non-Transactional
  - Memory (formerly HEAP) – Non-Transactional

NOTE: The DataDirect Connect Series *for* ODBC drivers for MySQL Enterprise were developed using the MySQL Protocol Documentation whose copyright is owned by, and licensed by DataDirect from, MySQL AB. If any of the DataDirect Connect Series *for* ODBC is licensed for the MySQL database the following shall apply: You must purchase commercially licensed MySQL database software or a MySQL Enterprise subscription in order to use the DataDirect Connect Series *for* ODBC drivers for MySQL Enterprise with MySQL software.

The MySQL Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

See the readme file shipped with your DataDirect Connect product for the file name of the MySQL Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Security
- Connection Pooling

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 350 and Table 8-1 on page 353 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 8-1 on page 353 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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

To configure a MySQL data source on Windows:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

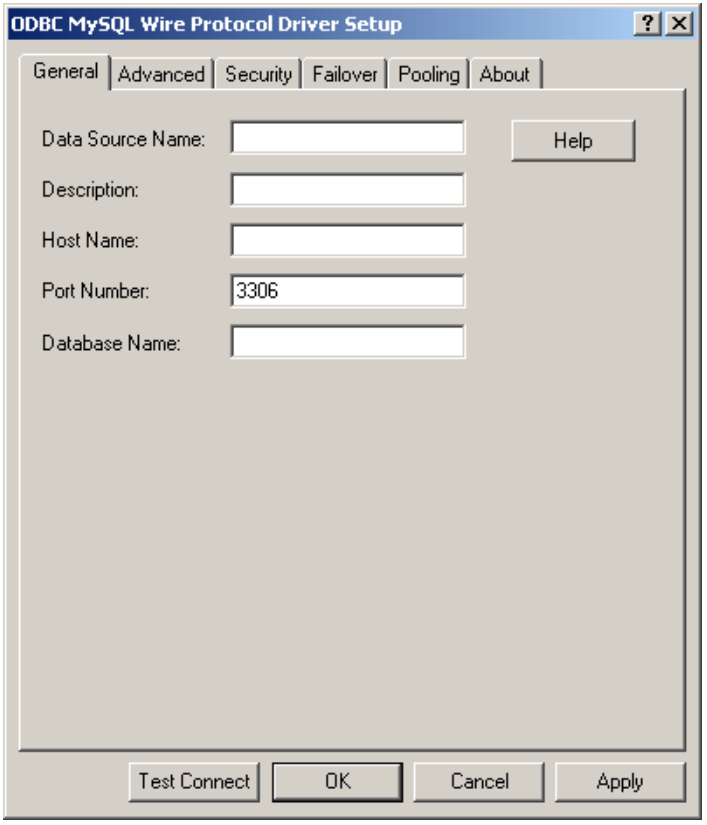
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

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

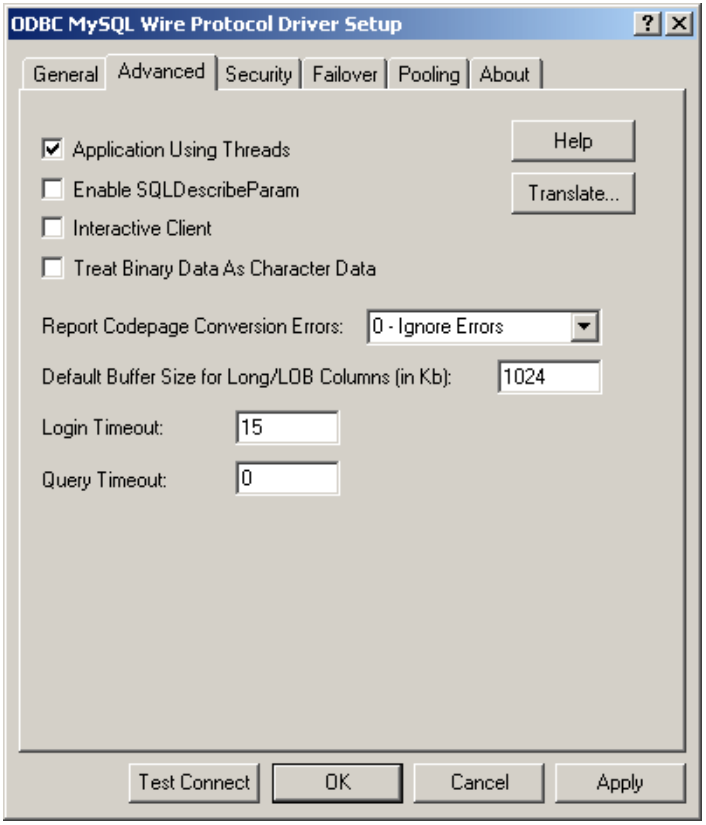


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 358)	None
Description (see page 359)	None
Host (see page 364)	None
Port Number (see page 372)	3306
Database Name (see page 358)	None

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



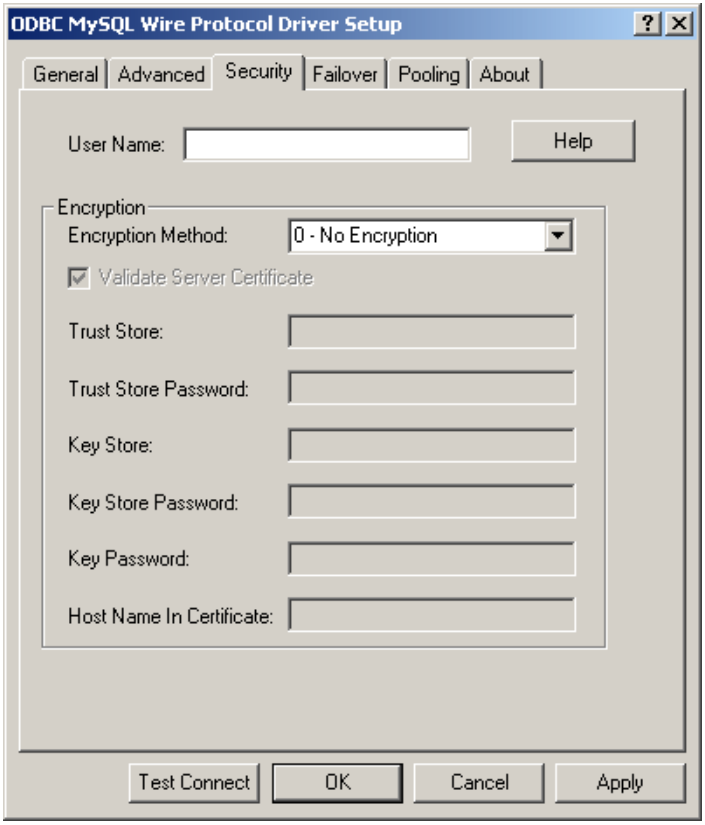
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Using Threads (see page 355)	Enabled
Enable SQLDescribeParam (see page 360)	Disabled
Interactive Client (see page 367)	Disabled
Treat Binary Data as Character Data (see page 374)	Disabled
Report Codepage Conversion Errors (see page 374)	0 - Ignore Errors
Default Buffer Size for Long/LOB Columns (in Kb) (see page 359)	1024
Login Timeout (see page 370)	15
Query Timeout (see page 373)	0
IANAAppCodePage (see page 366) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

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



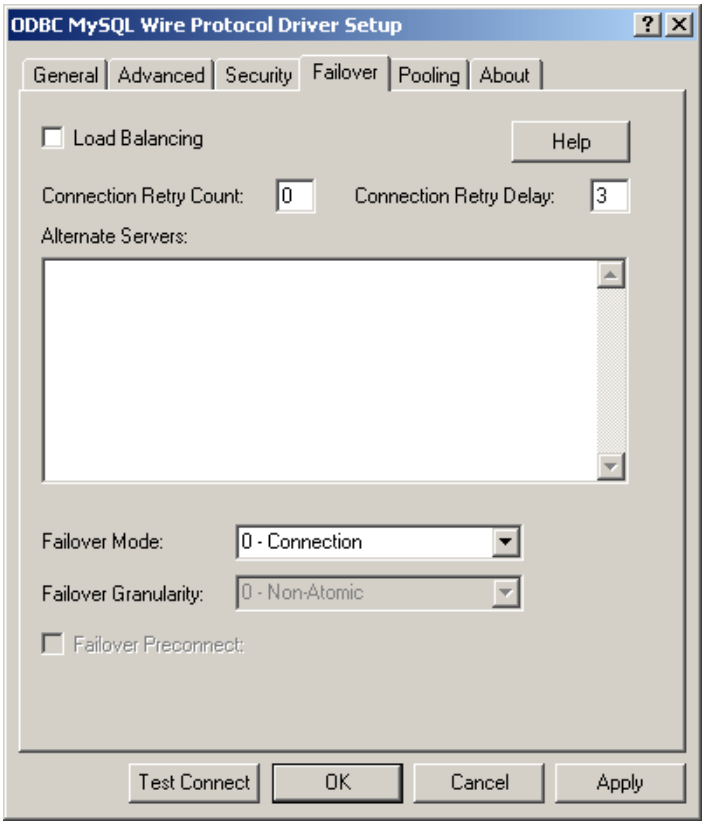
See "Using Security" on page 101 for a general description of authentication and encryption and their configuration requirements.

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

Connection Options: Security	Default
User Name (see page 376)	None
Encryption Method (see page 360)	0 - No Encryption
Validate Server Certificate (see page 377)	Enabled

Connection Options: Security	Default
Truststore (see page 375)	None
Truststore Password (see page 376)	None
Keystore (see page 368)	None
Keystore Password (see page 368)	None
Key Password (see page 367)	None
Host Name In Certificate (see page 364)	None

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

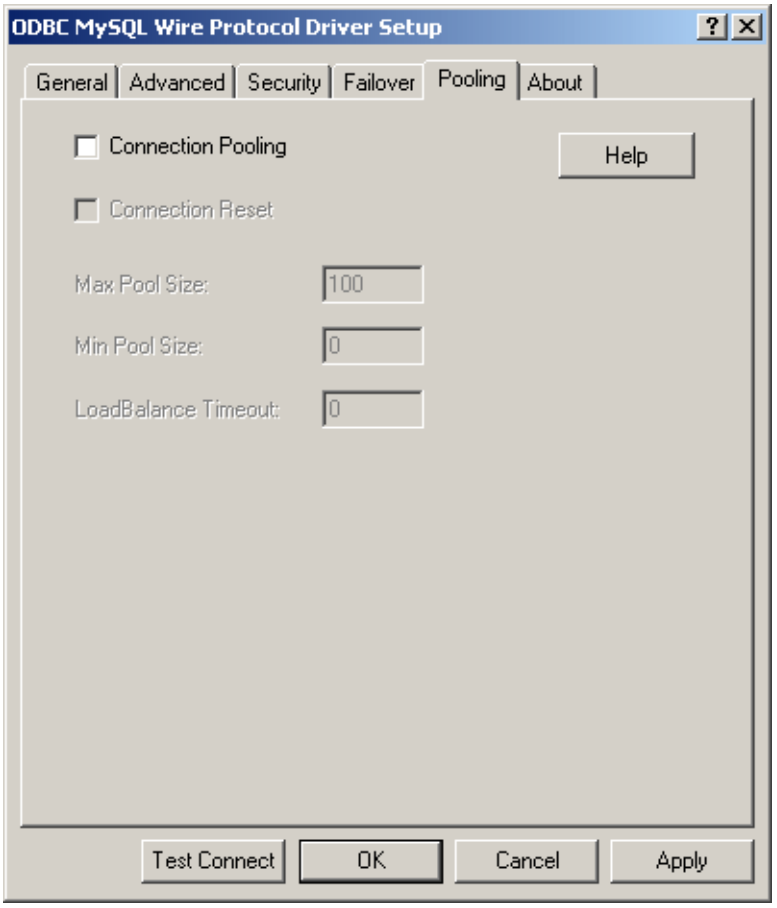


See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 369)	Disabled
Connection Retry Count (see page 357)	0
Connection Retry Delay (see page 357)	3
Alternate Servers (see page 354)	None
Failover Mode (see page 362)	0 - Connection
Failover Granularity (see page 361)	0 - Non-Atomic
Failover Preconnect (see page 363)	Disabled

7 Optionally, click the **Pooling** tab to specify connection pooling data source settings.



See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 355)	Disabled
Connection Reset (see page 356)	Disabled
Max Pool Size (see page 371)	100
Min Pool Size (see page 371)	0
Load Balance Timeout (see page 369)	0

8 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box. A logon dialog box appears; see "Using a Logon Dialog Box" on page 351 for details. Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an improper environment or incorrect connection value, it displays an appropriate error message.

  Click **OK**.

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

9 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 8-1 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 MySQL is:

```
DSN=MySQL TABLES;DB=PAYROLL
```

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

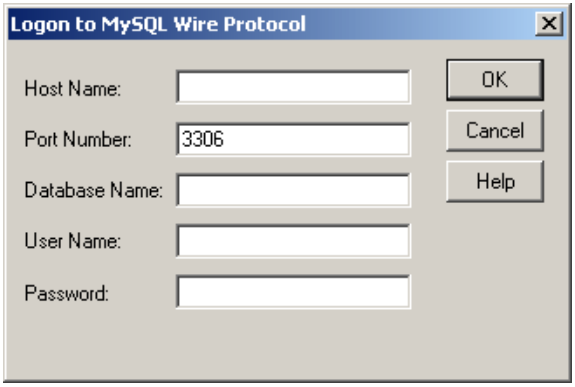
```
FILEDSN=MySQL.dsn;DB=DBPAYROLL
```

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

```
DRIVER=DataDirect MySQL Wire Protocol;
HOST=MySQL2;PORT=3306;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box

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



In this dialog box, perform the following steps:

1. In the Host Name field, type either the name or the IP address of the server to which you want to connect. The IP address must be in IPv4 format.
2. In the Port Number field, type the port number of the server listener. The default is 3306.
3. In the Database Name field, type the name of the database to which you want to connect.

4 If required, type your user name as specified on the MySQL server.

5 If required, type your password.

6 Click **OK** to complete the logon and to update these values in the Registry.

Connection Options Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

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

Table 8-1 lists the connection string attributes supported by the MySQL Wire Protocol driver.

Table 8-1. MySQL Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount	0
ConnectionRetryDelay	3
Database (DB)	None
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
Description (n/a)	None
EnableDescribeParam (EDP)	0 (Disabled)
EncryptionMethod (EM)	0 (Disabled)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InteractiveClient (IC)	0 (Disabled)
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoadBalanceTimeout (LBT)	0 (Disabled)
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)

Table 8-1. MySQL Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
PortNumber (PORT)	3306
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
TreatBinaryAsChar (TBAC)	0 (Disabled)
Truststore (TS)	None
TruststorePassword (TSP)	None
ValidateServerCertificate (VSC)	1 (Enabled)

Alternate Servers

Attribute AlternateServers (ASVR)

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

Valid Values `(Database=`*`databasename`*`:HostName=`*`hostvalue`*`:PortNumber=`*`portvalue`*`[, . . .])`

You must specify the database name, host name, and port number. The string has the format:

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

```
(Database=MySQLdb1:HostName=MySQLhost1:PortNumber=5177,
Database=MySQLdb2:HostName=MySQLhost2:PortNumber=5178)
```

Default None

GUI Tab Failover tab on page 346

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI tab Advanced tab on page 343

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 348

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 348

Connection Retry Count

Attribute ConnectionRetryCount

Description 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.

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 on page 346

Connection Retry Delay

Attribute ConnectionRetryDelay

Description 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.

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

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

Default 3

GUI Tab Failover tab on page 346

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 342

Database Name

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 342

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute DefaultLongDataBuffLen (DLDBL)

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

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values An integer in multiples of 1024

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

Default 1024

GUI tab Advanced tab on page 343

Description

Attribute Description (n/a)

Description 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 on page 342

Enable SQLDescribeParam

Attribute: EnableDescribeParam (EDP)

Description: Determines whether the driver uses the SQLDescribeParam function, which describes parameters as a data type of SQL_VARCHAR with a length of 255 for statements.

Valid Values: 0 | 1

If set to 1 (enabled), the SQLDescribeParam function describes parameters as a data type of SQL_VARCHAR with a length of 255 for statements.

If set to 0 (disabled), the SQLDescribeParam function returns the standard ODBC error IM001.

Default: 0 (Disabled)

GUI tab: Advanced tab on page 343

Encryption Method

Attribute: EncryptionMethod (EM)

Description: 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.

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values: 0 | 1 | 2

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

Default 0 (No Encryption)

GUI Tab Security tab on page 345

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 346

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

Valid Values 0 | 1 | 2

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 on page 346

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

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 on page 346

Host

Attribute: HostName (HOST)

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

Valid Values: *server_name* | *IP_address*

where

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

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

The IP address must be in IPv4 format.

Default: None

GUI Tab: General tab on page 342

Host Name In Certificate

Attribute: HostNameInCertificate (HNIC)

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

Valid Values *host_name* | #SERVERNAME#

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

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 Common Name parts.

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

Default None

GUI Tab Security tab on page 345

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

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

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

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

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI tab Advanced tab on page 343

Interactive Client

Attribute InteractiveClient (IC)

Description Determines how long a connection can be idle before the server disconnects it.

NOTE: The wait_timeout variable controlled by the Interactive Client option is a session variable that can be modified by the application after the connection has been established. The Interactive Client option controls only the initial value of the wait_timeout session variable.

Valid Values 0 | 1

If set to 1 (Enabled), the driver initializes the wait_time session variable for the connection with the value of the global interactive_timeout variable.

If set to 0 (Disabled), the driver initializes the wait_timeout session variable with the value of the global wait_timeout variable.

Default 0 (Disabled)

GUI tab Advanced tab on page 343

Key Password

Attribute KeyPassword (KP)

Description The password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values *key_password*

where *key_password* is the password of a key in the keystore.

Default None

GUI Tab Security tab on page 345

Keystore

Attribute Keystore (KS)

Description The directory of the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file.

Valid Values *keystore_directory*

where *keystore_directory* is the location of the keystore file.

Default None

GUI Tab Security tab on page 345

Keystore Password

Attribute KeystorePassword (KSP)

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

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

Valid Values *keystore_password*

where *keystore_password* is the password of the keystore file.

Default None

GUI Tab Security tab on page 345

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

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

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

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

Default 0 (Disabled)

GUI Tab Pooling tab on page 348

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 346

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

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

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI tab Advanced tab on page 343

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values An integer from 1 to 65535

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

Default 100

GUI Tab Pooling tab on page 348

Min Pool Size

Attribute MinPoolSize (MNPS)

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

This connection option can affect performance. See "Performance Considerations" on page 378 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

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

Default 0

GUI Tab Pooling tab on page 348

Password

Attribute Password (PWD)

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

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

NOTE: This option is mutually exclusive with the Server Name and TNSNames File options.

Valid Values *port_name*

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

Default None

GUI Tab General tab on page 342

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI tab Advanced tab on page 343

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 343

Treat Binary Data as Character Data

Attribute TreatBinaryAsChar (TBAC)

Description Allows data that MySQL stores as BINARY or VARBINARY to be described and returned as CHAR or VARCHAR values, respectively.

Valid Values 0 | 1

If set to 1 (Enabled), the driver describes and returns data that MySQL stores as BINARY or VARBINARY as CHAR or VARCHAR values, respectively.

If set to 0 (Disabled), the driver describes and returns data that MySQL describes as BINARY or VARBINARY as BINARY or VARBINARY values, respectively.

Example Create the following MySQL table:

```
CREATE TABLE binTable (col1 binary(3))
```

Then, execute the following Insert statement:

```
INSERT INTO binTable values('abc')
```

Then, execute the following query:

```
SELECT col1 FROM binTable
```

Using this example, the driver would return the value of col1 as a CHAR value, "abc", instead of a BINARY value "616263".

Default 0 (Disabled)

GUI tab Advanced tab on page 343

Truststore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 345

Truststore Password

Attribute TruststorePassword (TSP)

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

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

Valid Values *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default None

GUI Tab Security tab on page 345

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 on page 345

Validate Server Certificate

Attribute ValidateServerCertificate (VSC)

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

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

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Security tab on page 345

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for details.

The option names found on the tabs of the driver Setup dialog box are the same as the connection string attribute names unless otherwise noted in parentheses. The connection string attribute name does not have spaces between the words. For example, the option name Application Using Threads is equivalent to the connection string attribute name ApplicationUsingThreads.

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.

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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: 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.

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

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

Data Types

Table 8-2 shows how the MySQL data types map to the standard ODBC data types.

Table 8-2. MySQL Data Types

MySQL	ODBC
BIGINT	SQL_BIGINT
BIGINT UNSIGNED	SQL_BIGINT
BINARY	SQL_BINARY
BIT	SQL_BINARY
BLOB	SQL_LONGVARBINARY
CHAR	SQL_CHAR
DATE	SQL_TYPE_DATE
DATETIME	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
DECIMAL UNSIGNED	SQL_DECIMAL
DOUBLE	SQL_DOUBLE
DOUBLE UNSIGNED	SQL_DOUBLE
FLOAT	SQL_REAL
FLOAT UNSIGNED	SQL_REAL
INTEGER	SQL_INTEGER
INTEGER UNSIGNED	SQL_INTEGER
LONGBLOB	SQL_LONGVARBINARY
LONGTEXT	SQL_LONGVARCHAR
MEDIUMBLOB	SQL_LONGVARBINARY
MEDIUMINT	SQL_INTEGER
MEDIUMINT UNSIGNED	SQL_INTEGER
MEDIUMTEXT	SQL_LONGVARCHAR
SMALLINT	SQL_SMALLINT
SMALLINT UNSIGNED	SQL_SMALLINT

Table 8-2. MySQL Data Types (cont.)

MySQL	ODBC
TEXT	SQL_LONGVARCHAR
TIME	SQL_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP
TINYBLOB	SQL_LONGVARBINARY
TINYINT	SQL_TINYINT
TINYINT UNSIGNED	SQL_TINYINT
TINYTEXT	SQL_LONGVARCHAR
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_VARCHAR
YEAR	SQL_SMALLINT

Refer to "Retrieving Data Type Information" in Chapter 2 "Using The Product" of the *DataDirect Connect Series for ODBC User's Guide* for information about retrieving data types.

NOTE: The Treat Binary Data as Character Data connection connection option affects how certain ODBC data types are reported. See "Treat Binary Data as Character Data" on page 374 for details.

Unicode Support

When the character set of a character column is Unicode, then the MySQL Wire Protocol driver maps the MySQL data type to Unicode data type as follows:

MySQL Data Type	Mapped to. . .
CHAR	SQL_WCHAR

MySQL Data Type	Mapped to. . .
LONGTEXT	SQL_WLONGVARCHAR
MEDIUMTEXT	SQL_WLONGVARCHAR
TEXT	SQL_WLONGVARCHAR
TINYTEXT	SQL_WLONGVARCHAR
VARCHAR	SQL_WVARCHAR

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

MySQL supports isolation levels 0 (read uncommitted), 1 (read committed), 2 (repeatable read), and 3 (serializable). The default is 1.

MySQL supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" of the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The MySQL Wire Protocol driver supports the following functions:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" of the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the MySQL Wire Protocol driver.

Number of Connections and Statements Supported

The MySQL Wire Protocol driver supports multiple connections and multiple statements per connection to the MySQL database system.

9 The Oracle Wire Protocol Driver

The DataDirect Connect Series *for* ODBC Oracle Wire Protocol driver (the Oracle Wire Protocol driver) is available in both 32- and 64-bit versions and supports the following Oracle database servers:

- Oracle 11*g* R1 (11.1)
- Oracle 10*g* R1 and R2 (10.1 and 10.2)
- Oracle 9*i* R1 and R2 (9.0.1 and 9.2)
- Oracle 8*i* R2 and R3 (8.1.6 and 8.1.7)

The Oracle Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the Windows, UNIX, and Linux environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Oracle Wire Protocol driver.

NOTE: The Oracle Wire Protocol driver does not require any Oracle client software. DataDirect Technologies also provides an Oracle client-based driver; see "The Oracle Driver" on page 695 for details.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Security
- Connection Pooling
- DataDirect Bulk Load

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Kerberos Authentication Requirements

Verify that your environment meets the requirements listed in Table 9-1 before you configure the driver for Kerberos authentication.

Table 9-1. Kerberos Authentication Requirements for the Oracle Wire Protocol Driver

Component	Requirements
Database server	The database server must be administered by the same domain controller that administers the client and must be running one of the following databases: ■ Oracle 11*g* (R1 and R2) ■ Oracle 10*g* (R1 and R2) ■ Oracle 9*i* (R2) In addition, Oracle Advanced Security is required.

Table 9-1. Kerberos Authentication Requirements for the Oracle Wire Protocol Driver *(cont.)*

Component	Requirements
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by one of the following methods: ■ Windows Active Directory on one of the following operating systems: Windows Server 2003 or Windows 2000 Server Service Pack 3 or higher ■ MIT Kerberos 1.4.2 or higher
Client	The client must be administered by the same domain controller that administers the database server.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 “Quick Start Connect” on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 412 and Table 9-2 on page 416 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 9-2 on page 416 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

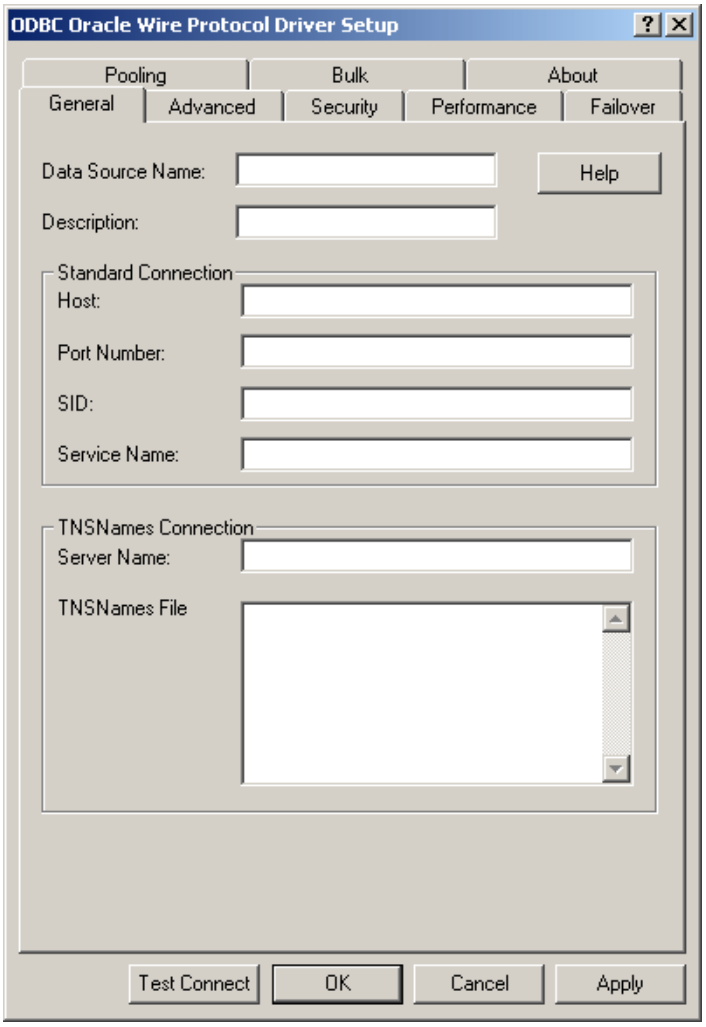
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

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

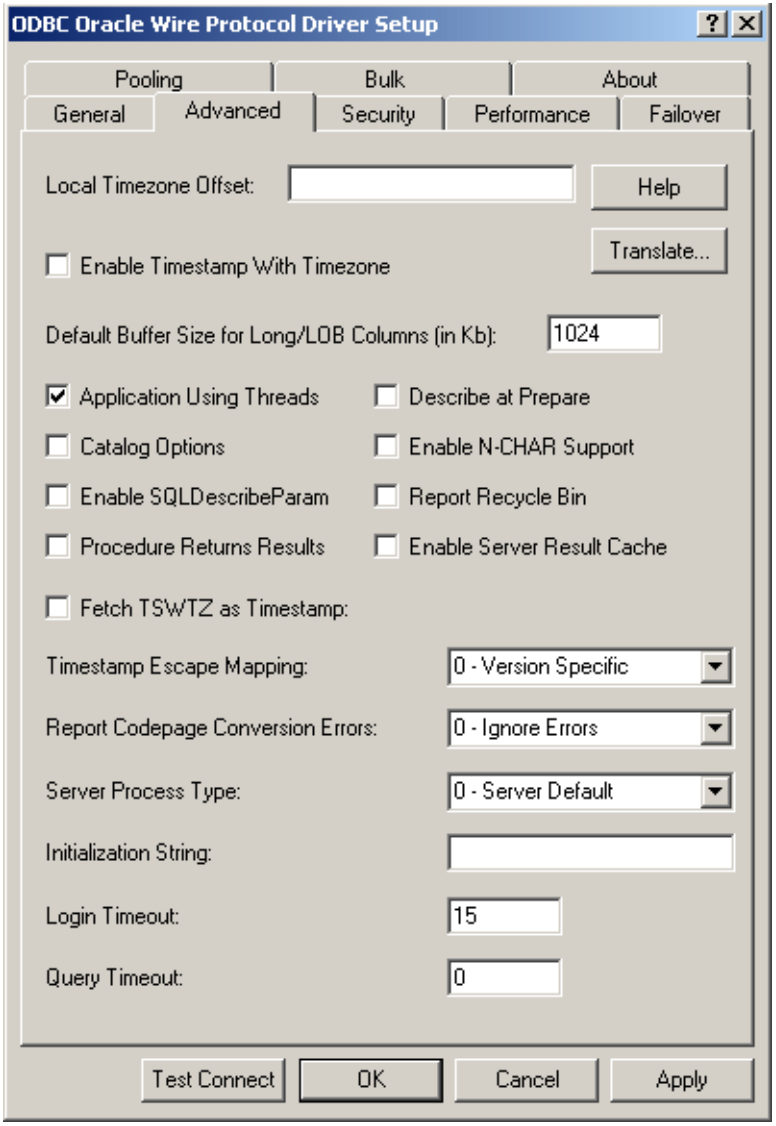


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 428)	None
Description (see page 430)	None
Host (see page 439)	None
Port Number (see page 449)	None
SID (see page 455)	None
Service Name (see page 454)	None
Server Name (see page 452)	None
TNSNames File (see page 456)	None

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



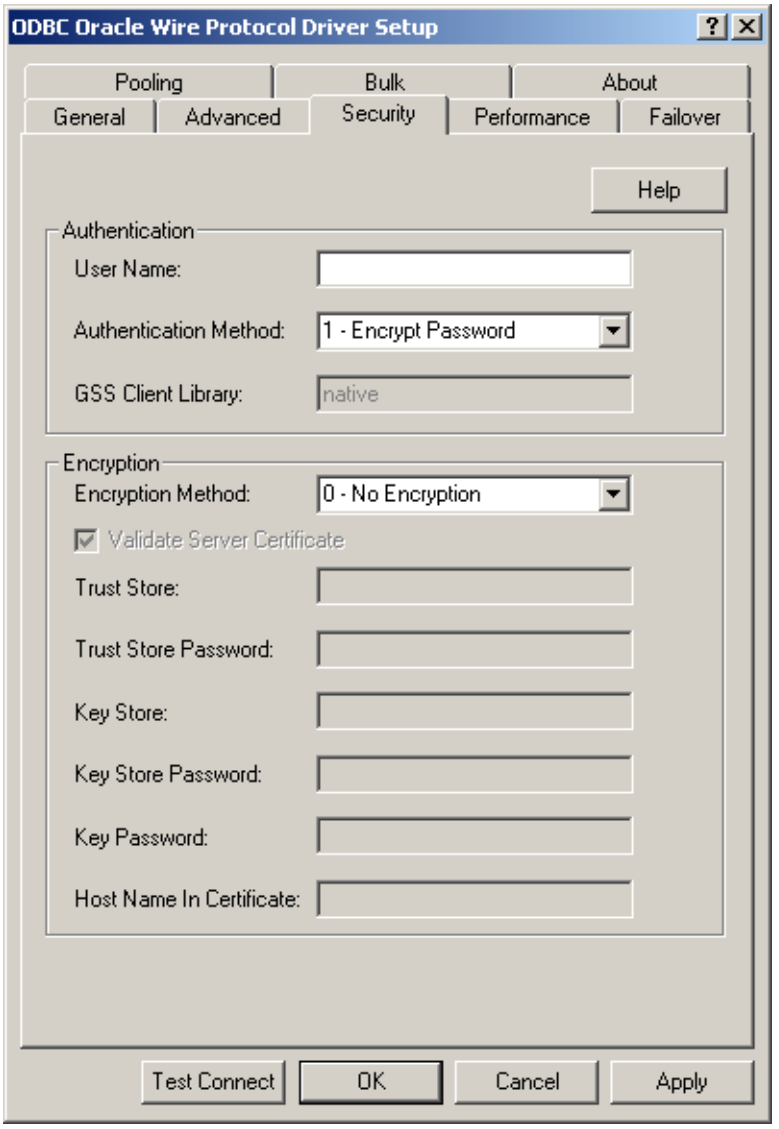
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Local Timezone Offset (see page 445)	None
Enable Timestamp with Timezone (see page 434)	Disabled
Default Buffer Size for Long/LOB Columns (in Kb) (see page 428)	1024
Application Using Threads (see page 419)	Enabled
Catalog Options (see page 425)	Disabled
Enable SQLDescribeParam (see page 432)	Disabled
Procedure Returns Results (see page 449)	Disabled
Describe at Prepare (see page 429)	Disabled
Enable N-CHAR Support (see page 430)	Disabled
Report Recycle Bin (see page 452)	Disabled
Enable Server Result Cache (see page 432)	Disabled
Fetch TSWTZ as Timestamp (see page 438)	Disabled
Timestamp Escape Mapping (see page 455)	0 - Version Specific
Report Codepage Conversion Errors (see page 451)	0 - Ignore Errors
Server Process Type (see page 453)	0 - Server Default
Initialization String (see page 442)	None
Login Timeout (see page 446)	15
Query Timeout (see page 450)	0
IANAAppCodePage (see page 441) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

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



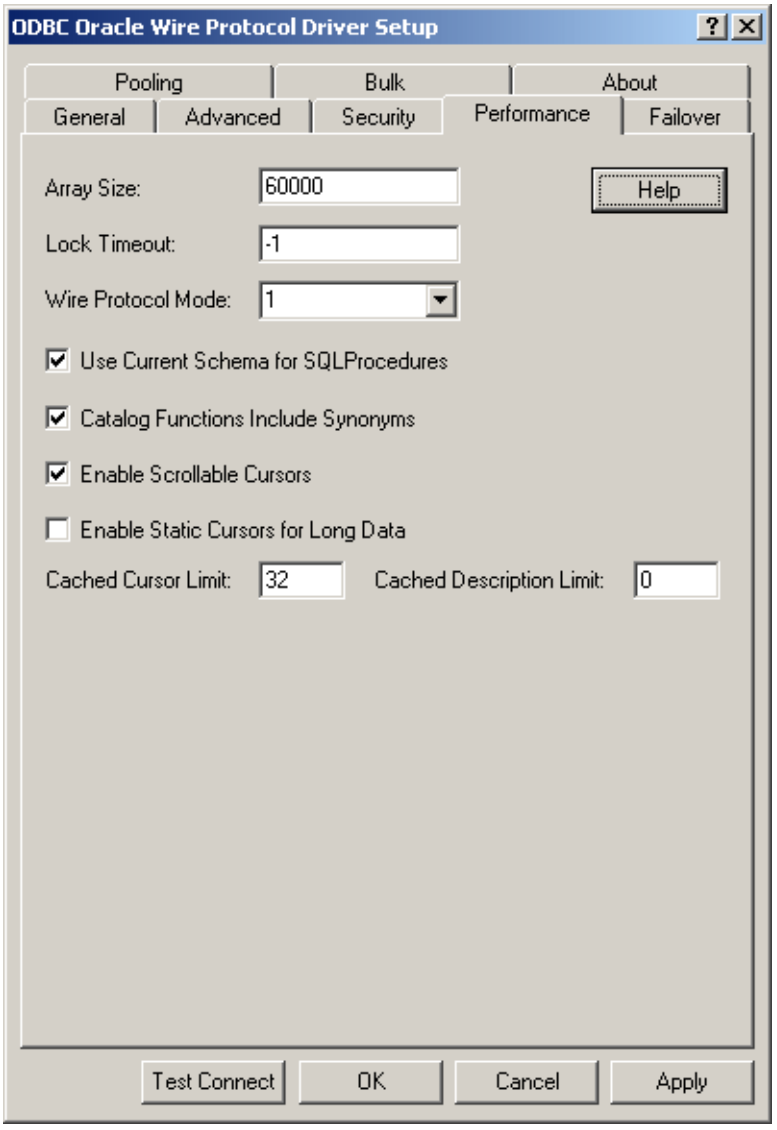
See "Using Security" on page 101 for a general description of authentication and encryption and their configuration requirements.

See "OS Authentication" on page 473 for a discussion of Oracle and SLL encryption.

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

Connection Options: Security	Default
User Name (see page 459)	None
Authentication Method (see page 420)	1 - Encrypt Password
GSS Client Library (see page 438)	native
Encryption Method (see page 434)	0 - No Encryption
Validate Server Certificate (see page 459)	None
Truststore (see page 457)	None
Truststore Password (see page 458)	None
Keystore (see page 443)	None
Keystore Password (see page 443)	None
Key Password (see page 442)	None
Host Name In Certificate (see page 440)	None

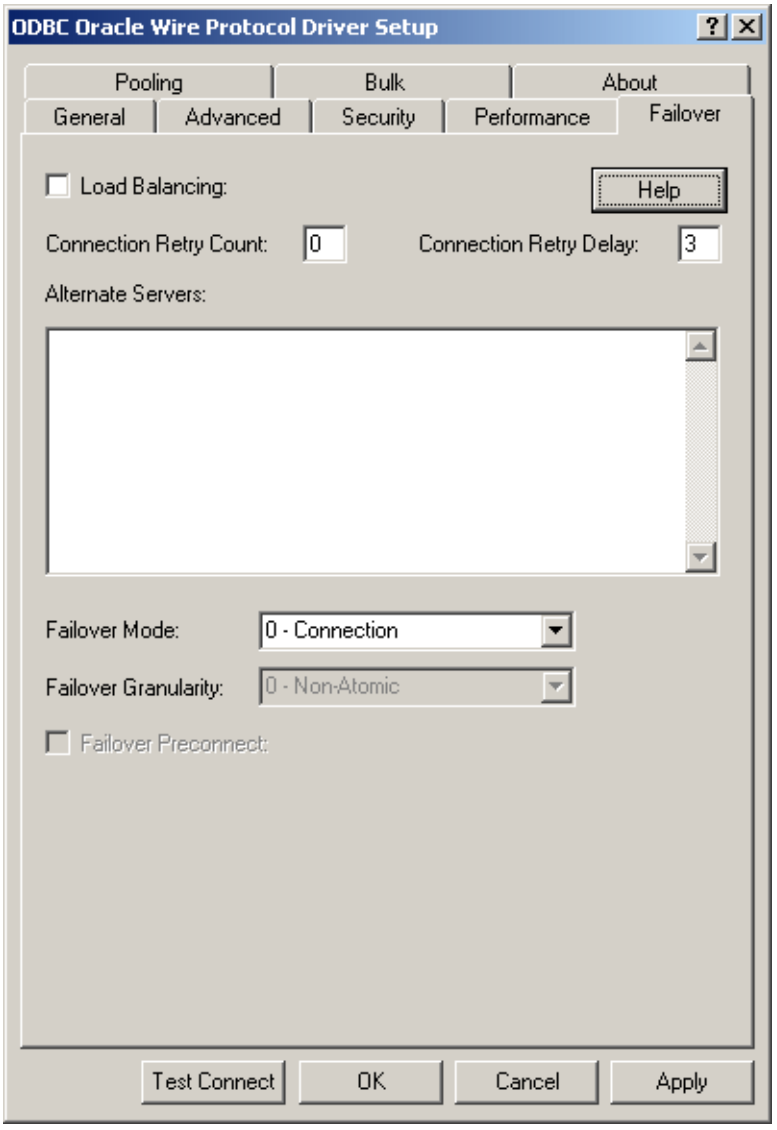
6 Optionally, click the **Performance** tab to specify performance data source settings.



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

Connection Options: Performance	Default
Array Size (see page 420)	60000
Lock Timeout (see page 446)	-1
Wire Protocol Mode (see page 460)	1
Use Current Schema for SQLProcedures (see page 458)	Enabled
Catalog Functions Include Synonyms (see page 424)	Enabled
Enable Scrollable Cursors (see page 431)	Enabled
Enable Static Cursors for Long Data (see page 433)	Disabled
Cached Cursor Limit (see page 423)	32
Cached Description Limit (see page 424)	0

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

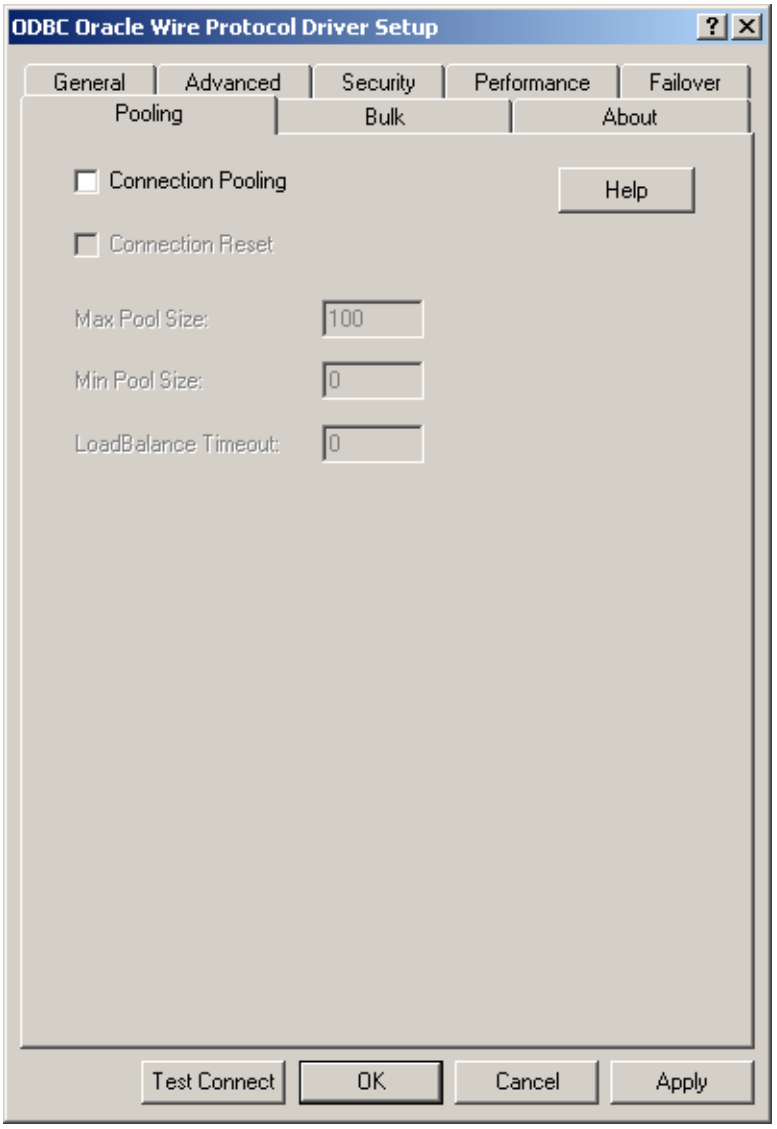


See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 444)	Disabled
Connection Retry Count (see page 427)	0
Connection Retry Delay (see page 427)	3
Alternate Servers (see page 418)	None
Failover Mode (see page 436)	0 - Connection
Failover Granularity (see page 435)	0 - Non-Atomic
Failover Preconnect (see page 437)	Disabled

8 Optionally, click the **Pooling** tab to specify connection pooling data source settings.

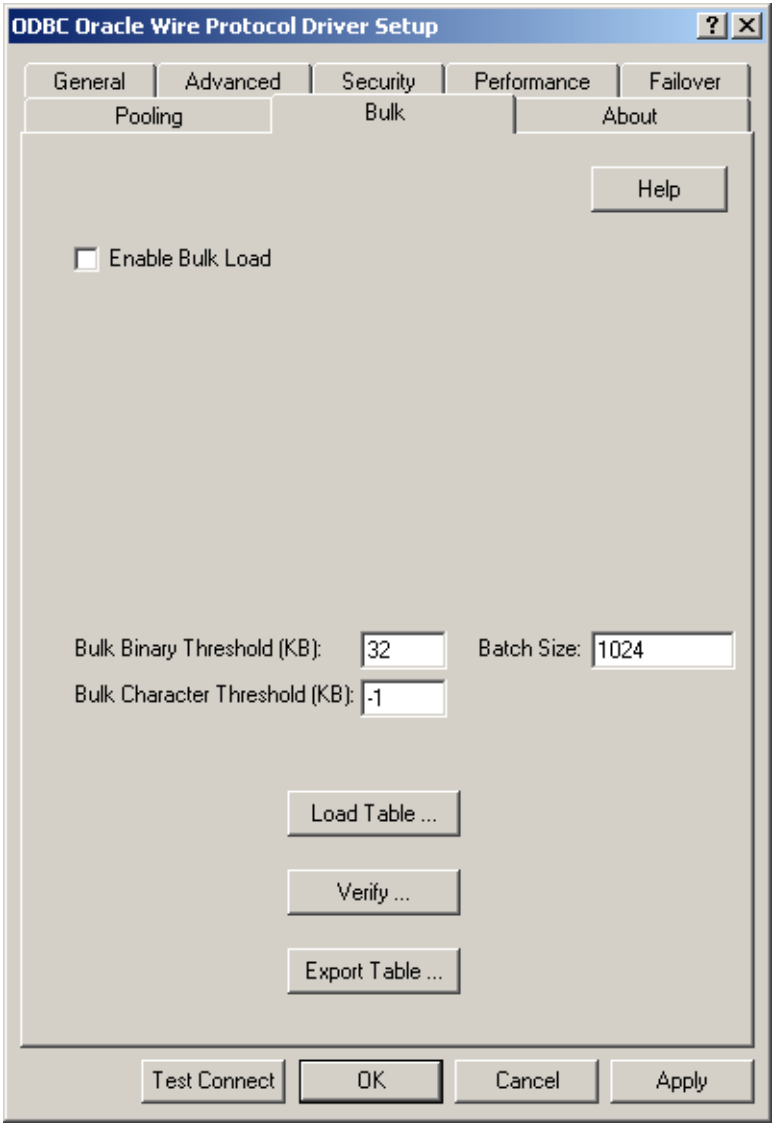


See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 425)	Disabled
Connection Reset (see page 426)	Disabled
Max Pool Size (see page 447)	100
Min Pool Size (see page 448)	0
Load Balance Timeout (see page 444)	0

9 Optionally, click the **Bulk** tab to specify DataDirect Bulk Load data source settings.



See "Using DataDirect Bulk Load" on page 115 for a general description of DataDirect Bulk Load.

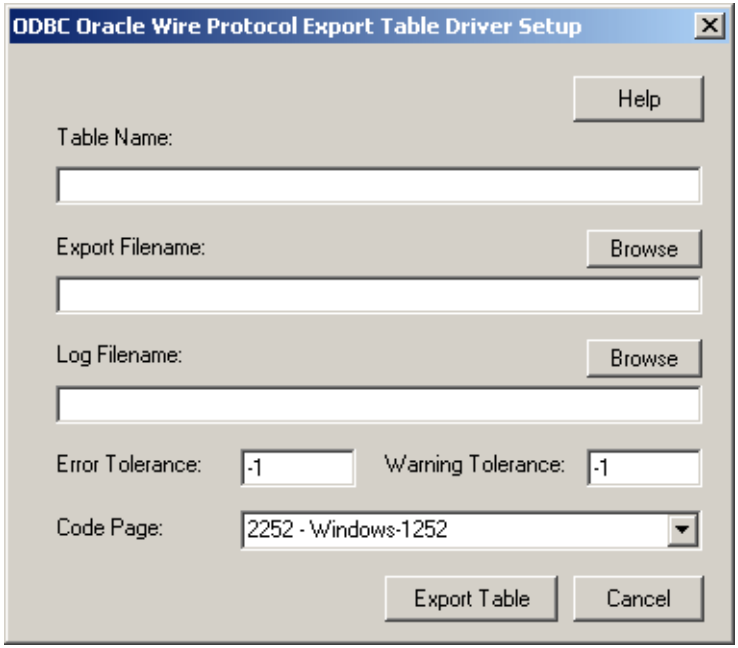
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Bulk	Default
Enable Bulk Load (see page 430)	Disabled
Bulk Binary Threshold (see page 422)	32
Bulk Character Threshold (see page 422)	-1
Batch Size (see page 421)	1024

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

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

10 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.



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" on page 115 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" on page 129 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. This file must not already exist; if the file already exists, an error is returned.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. Events logged to this file are:

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

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

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

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

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

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

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

Code Page: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See "Character Set Conversions" on page 128 for more information.

The default value on Windows is the current code page of the machine. On UNIX/Linux, the default value is 4.

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

11 To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify** from the Bulk tab. See "Verification of the Bulk Load Configuration File" on page 125 for details. The Verify dialog box appears.

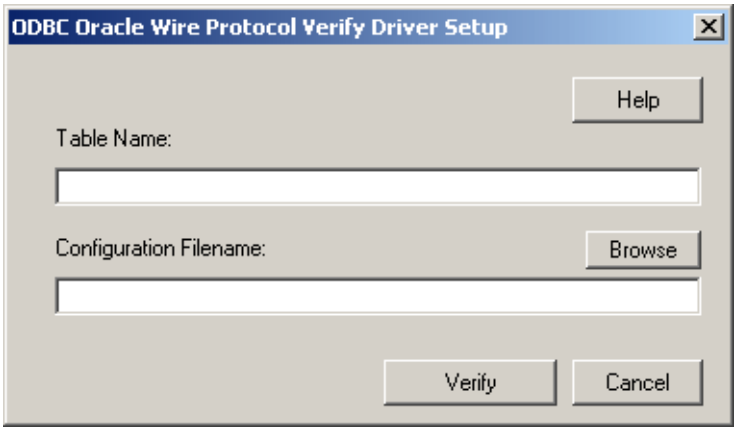
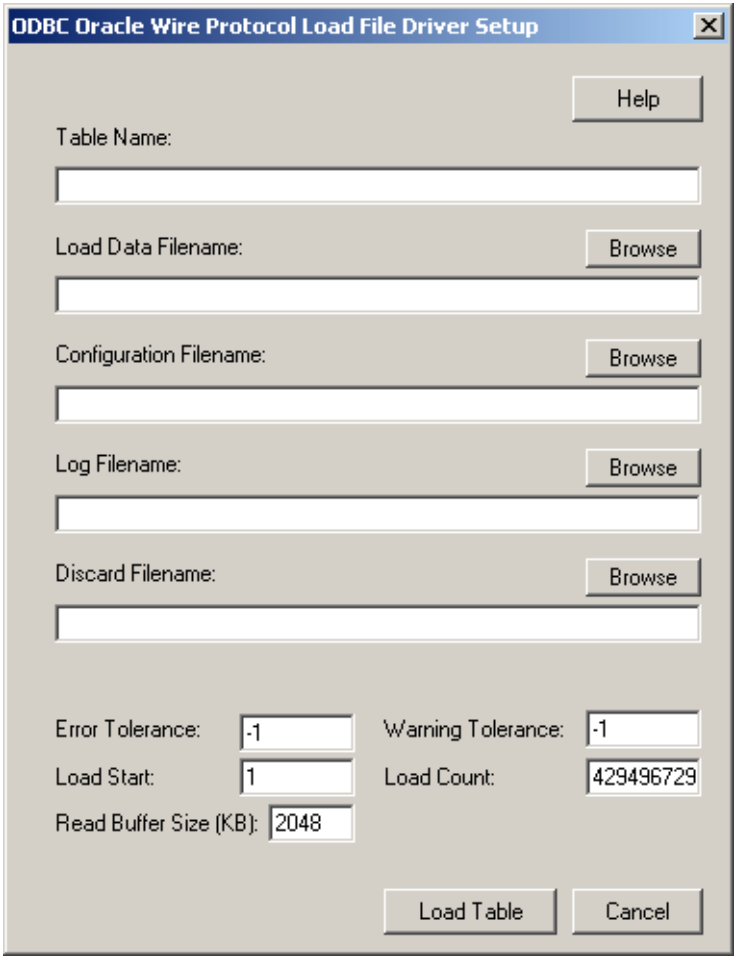


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

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

13 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.



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 to be 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 to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. Supplying 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 supply 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. 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 you do not supply a value for Discard Filename, no discard 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.

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 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 4294967295.

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

14 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" on page 413 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

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

15 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 9-2 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:

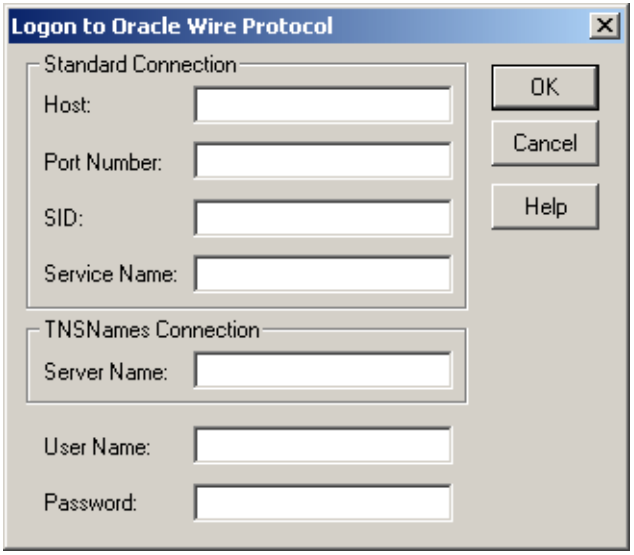
```
FILEDSN=OracleWP.dsn;ID=JOHN;PWD=XYZZY
```

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

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

Using a Logon Dialog Box

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



In this dialog box, provide the following information:

NOTE: To configure a standard connection, complete the first four fields and skip to Step 6.

1. In the Host field, type either the name or the IP address of the server to which you want to connect. The IP address must be in IPv4 format.

   If you enter a value for this field, the Server Name field is not available.

   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.

   If you enter a value for this field, the Server Name field is not available.

   This field is not available if you enter a value for the Server Name field.

3. In the SID field, type the Oracle System Identifier that refers to the instance of Oracle running on the server.

   If you enter a value for this field, the Server Name and Service Name fields are not available.

   This field is not available if you enter a value for the Service Name or Server Name fields.

4. In the Service Name field, type the Oracle service name that specifies the database used for the connection.

   See Service Name under Step 3 in "Configuring and Connecting to Data Sources" on page 387 for details.

   If you enter a value for this field, the Server Name and SID fields are not available.

   This field is not available if you enter a value for the SID or Server Name field.

NOTE: If you want to configure a TNSNames connection, complete only the following two fields.

5 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.

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.

6 If required, type your Oracle user name.

7 If required, type your Oracle password.

8 Click **OK** to log on to the Oracle database installed on the server you specified and to update the values in the Registry.

NOTE: You can also use OS Authentication to connect to an Oracle database. See "OS Authentication" on page 473 for details.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an

option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

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

Table 9-2 lists the connection string attributes supported by the Oracle Wire Protocol driver.

Table 9-2. Oracle Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	60000
AuthenticationMethod (AM)	1 (Encrypt Password)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadBatchSize (BLBS)	1024
CachedCursorLimit (CCL)	32
CachedDescriptionLimit (CDL)	0
CatalogIncludesSynonyms (CIS)	1 (Enabled)
CatalogOptions (CO)	0 (Disabled)
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
DescribeAtPrepare (DAP)	0 (Disabled)
Description (n/a)	None
EnableBulkLoad (EBL)	0 (Disabled)
EnableDescribeParam (EDP)	0 (Disabled)
EnableNcharSupport (ENS)	0 (Disabled)

Table 9-2. Oracle Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
EnableScrollableCursors (ESC)	1 (Enabled)
EnableServerResultCache (ESRC)	0 (Disabled)
EnableStaticCursorsForLongData (ESCLD)	0 (Disabled)
EnableTimestampwithTimezone (ETWT)	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
GSSClientLibrary (GS)	native
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoadBalanceTimeout (LBT)	0 (Disabled)
LoadBalancing (LB)	0 (Disabled)
LocalTimezoneOffset (LTZO)	"" (Empty String)
LockTimeout (LTO)	-1
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
PortNumber (PORT)	None
ProcedureRetResults (PRR)	0 (Disabled)

Table 9-2. Oracle Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
ReportRecycleBin (RRB)	0 (Disabled)
ServerName (SRVR)	None
ServerType (ST)	0 (Server Default)
ServiceName (SN)	None
SID (SID)	None
TimestampEscapeMapping (TEM)	0 (Oracle Version Specific)
TNSNamesFile (TNF)	None
Truststore (TS)	None
TruststorePassword (TSP)	None
UseCurrentSchema (UCS)	1 (Enabled)
ValidateServerCertificate (VSC)	1 (Enabled)
WireProtocolMode (WPM)	1

Alternate Servers

Attribute AlternateServers (ASVR)

Description A list of alternate database servers to which the driver will try 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 on page 399

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI tab Advanced tab on page 393

Array Size

Attribute ArraySize (AS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

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.

Default 60000

GUI Tab Performance tab on page 397

Authentication Method

Attribute AuthenticationMethod (AM)

Description 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

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), 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.

Default

1 (Encrypt Password)

GUI tab Security tab on page 395

Batch Size

Attribute BulkLoadBatchSize (BLBS)

Description The number of rows at a time that the driver sends to the database during bulk operations. This value applies to all methods of bulk loading.

Valid Values 0 | *x*

where *x* is the number of rows to send during a bulk operation.

Default 1024

GUI Tab Bulk tab on page 403

Bulk Binary Threshold

Attribute BulkBinaryThreshold (BBT)

Description 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.

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 on page 403

Bulk Character Threshold

Attribute BulkCharacterThreshold (BCT)

Description 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.

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 on page 403

Cached Cursor Limit

Attribute CachedCursorLimit (CCL)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values An integer from 0 to 65535

Default 32

GUI Tab Performance tab on page 397

Cached Description Limit

Attribute: CachedDescriptionLimit (CDL)

Description: 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.

NOTE: If the Select statement contains a Union or a nested Select, the description is not cached.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values: An integer from 0 to 65535

Default: 0

GUI Tab: Performance tab on page 397

Catalog Functions Include Synonyms

Attribute: CatalogIncludesSynonyms (CIS)

Description: Determines whether synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values: 0 | 1

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.

Default 1 (Enabled)

GUI Tab Performance tab on page 397

Catalog Options

Attribute CatalogOptions (CO)

Description Determines whether SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 401

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 401

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 399

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

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

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

Default 3

GUI Tab Failover tab on page 399

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 392

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute DefaultLongDataBuffLen (DLDBL)

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

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values An integer in multiples of 1024

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

Default 1024

GUI Tab Advanced tab on page 393

Describe at Prepare

Attribute DescribeAtPrepare (DAP)

Description Determines whether the driver describes the SQL statement at prepare time.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

Description

Attribute Description (n/a)

Description 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 on page 392

Enable Bulk Load

Attribute EnableBulkLoad (EBL)

Description Specifies the bulk load method.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses the database bulk load protocols. If the protocols cannot be used, the driver returns an error.

If set to 0 (Disabled), the driver uses standard parameter arrays.

Default 0 (Disabled)

GUI Tab Bulk tab on page 403

Enable N-CHAR Support

Attribute EnableNcharSupport (ENS)

Description 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.

See "Unicode Support" on page 470 for details.

NOTE: Valid only on Oracle 9*i* and higher.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

Enable Scrollable Cursors

Attribute EnableScrollableCursors (ESC)

Description Determines whether scrollable cursors, both Keyset and Static, are enabled for the data source.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

If set to 1 (Enabled), scrollable cursors are enabled for the data source.

If set to 0 (Disabled), scrollable cursors are not enabled.

Default 1 (Enabled)

GUI Tab Performance tab on page 397

Enable Server Result Cache

Attribute EnableServerResultCache (ESRC)

Description Determines whether the driver adds a hint to SQL statements to enable Oracle's server-side resultset caching feature, which stores the result set in database memory so that it can be reused. Server-side resultset caching can improve performance if your application executes the same query multiple times.

This option only applies to connections to Oracle 11*g* database servers that support server-side resultset caching.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver adds a hint to SQL statements to enable server-side resultset caching. For example:

```
SELECT /*+ result_cache */ * FROM employees
```

If set to 0 (Disabled), the driver does not add a hint to SQL statements.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

Enable SQLDescribeParam

Attribute EnableDescribeParam (EDP)

Description Determines whether the SQLDescribeParam function describes all parameters with a data type of SQL_VARCHAR for Select statements. For Insert/Update/Delete statements and for stored procedures, the parameters are described as the actual Oracle

data types on the Oracle server. This option must be enabled to access data when using Microsoft Remote Data Objects (RDO).

Valid Values 0 | 1

If set to 1 (Enabled), the SQLDescribeParam function describes all parameters with a data type of SQL_VARCHAR for Select statements.

If set to 0 (Disabled), the SQLDescribeParam function does not describe all parameters with a data type of SQL_VARCHAR for Select statements.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

Enable Static Cursors for Long Data

Attribute EnableStaticCursorsForLongData (ESCLD)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

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.

NOTE: You must enable this option if you want to persist a result set that contains Long data into an XML data file.

Default 0 (Disabled)

GUI Tab Performance tab on page 397

Enable Timestamp with Timezone

Attribute: EnableTimestampwithTimezone (ETWT)

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

Valid Values: 0 | 1

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: 0 (Disabled)

GUI Tab: Advanced tab on page 393

Encryption Method

Attribute: EncryptionMethod (EM)

Description: 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.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values: 0 | 1 | 3 | 4 | 5

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

If set to 3 (SSL3), the driver uses SSL3 data encryption.

If set to 4 (SSL2), the driver uses SSL2 data encryption.

If set to 5 (TLS1), the driver uses TLS1 data encryption.

NOTE: Consult your database administrator concerning the SSL settings of your Oracle server.

Default 0 (No Encryption)

GUI Tab Security tab on page 395

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 399

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

Valid Values 0 | 1 | 2

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 on page 399

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

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 on page 399

Fetch TSWTZ as Timestamp

Attribute: FetchTSWTZasTimestamp (FTSWTZAT)

Description: Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid on Oracle 10*g* R2 or higher.

Valid Values: 0 | 1

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 timezone 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 timezone information in the fetched value.

Default: 0 (Disabled)

GUI Tab: Advanced tab on page 393

GSS Client Library

Attribute: GSSClientLibrary (GS)

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

The driver uses the standard path for loading the specified client library.

Valid Values: native | *client_library*

where *client_library* is a GSS client library installed on the client.

If set to *client_library*, the driver uses the specified GSS client library.

If set to native, the driver uses the GSS client shipped with the operating system.

Default native

GUI Tab Security tab on page 395

Host

Attribute HostName (HOST)

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

Valid Values *server_name* | *IP_address*

where

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

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

The IP address must be in IPv4 format.

NOTE: This option is mutually exclusive with the Server Name and TNSNames File options.

Default None

GUI Tab General tab on page 392

Host Name In Certificate

Attribute HostNameInCertificate (HNIC)

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

Valid Values *host_name* | #SERVERNAME#

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

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 Common Name parts.

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

Default None

GUI Tab Security tab on page 395

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

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

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

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

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 393

Initialization String

Attribute InitializationString (IS)

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

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

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'
```

Default None

GUI Tab Advanced tab on page 393

Key Password

Attribute KeyPassword (KP)

Description The password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values *key_password*

where *key_password* is the password of a key in the keystore.

Default None

GUI Tab Security tab on page 395

Keystore

Attribute Keystore (KS)

Description The directory of the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file.

Valid Values *keystore_directory*

where *keystore_directory* is the location of the keystore file.

Default None

GUI Tab Security tab on page 395

Keystore Password

Attribute KeystorePassword (KSP)

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

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

Valid Values *keystore_password*

where *keystore_password* is the password of the keystore file.

Default None

GUI Tab Security tab on page 395

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

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

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

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

Default 0 (Disabled)

GUI Tab Pooling tab on page 401

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 399

Local Timezone Offset

Attribute LocalTimezoneOffset (LTZO)

Description 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 on page 393

Lock Timeout

Attribute LockTimeout (LTO)

Description Specifies the amount of time, in seconds, the Oracle server waits for a lock to be released before generating an error when processing a Select...For Update statement on an Oracle 9*i* or higher server.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values -1 | 0 | *x*

where *x* is an integer that specifies a number of seconds.

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.

NOTE: If you are connected to an Oracle 8*i* server, any value greater than 0 is equivalent to the value -1.

Default -1

GUI Tab Performance tab on page 397

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

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

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 393

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values An integer from 1 to 65535

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

Default 100

GUI Tab Pooling tab on page 401

Min Pool Size

Attribute MinPoolSize (MNPS)

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

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

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

Default 0

GUI Tab Pooling tab on page 401

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data

source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

NOTE: This option is mutually exclusive with the Server Name and TNSNames File options.

Valid Values *port_name*

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

Default None

GUI Tab General tab on page 392

Procedure Returns Results

Attribute ProcedureRetResults (PRR)

Description Determines whether the driver returns result sets from stored procedures/functions.

See "Support of Materialized Views" on page 474 for details.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 393

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI Tab Advanced tab on page 393

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 393

Report Recycle Bin

Attribute ReportRecycleBin (RRB)

Description Determines whether support is provided for reporting objects that are in the Oracle Recycle Bin.

On Oracle 10*g* R1 and higher, when a table is dropped, it is not actually removed from the database, but placed in the recycle bin instead.

Valid Values 0 | 1

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 on page 393

Server Name

Attribute ServerName (SRVR)

Description 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.

NOTE: This option is mutually exclusive with the Host, Port Number, SID, and Service Name options.

Valid Values *server_name*

where *server_name* is a net service name in the TNSNAMES.ORA file.

Default None

GUI Tab General tab on page 392

Server Process Type

Attribute ServerType (ST)

Description Determines whether the connection is established using a shared or dedicated server process (dedicated thread on Windows).

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 0 | 1 | 2

If set to 0 (Server Default), the driver uses the default server process set on the server.

NOTE: 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.

If set to 1 (Shared), the server process used is retrieved from a pool. The socket connection between the application and server is made to a dispatcher process on the server. This setting allows there to be fewer processes than the number of connections, reducing the need for server resources. Use this value when a server must handle a large number of connections.

If set to 2 (Dedicated), a server process is created to service only that connection. When that connection ends, so does the process (UNIX and Linux) or thread (Windows). The socket connection is made directly between the application and the dedicated server process or thread. When connecting to UNIX and Linux servers, a dedicated server process can provide significant performance improvement, but uses more resources on the server. When connecting to Windows servers, the server resource penalty is

insignificant. Use this value if you have a batch environment with a low number of connections.

Default 0 (Server Default)

GUI Tab Advanced tab on page 393

Service Name

Attribute ServiceName (SN)

Description 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.

This option is mutually exclusive with the SID, Server Name, and TNSNames File options.

Valid Values *`service_name`*

where *`service_name`* is the description of the destination for a network connection.

Default None

GUI Tab General tab on page 392

SID

Attribute SID (SID)

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

NOTE: This option is mutually exclusive with the Service Name, Server Name, and TNSNames File options.

Valid Values *sid*

where *sid* is the name of the Oracle System Identifier.

Default None

GUI Tab General tab on page 392

Timestamp Escape Mapping

Attribute TimestampEscapeMapping (TEM)

Description Determines how the driver maps Date, Time, and Timestamp literals.

Valid Values 0 | 1

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 on page 393

TNSNames File

Attribute TNSNamesFile (TNF)

Description 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.

NOTE: This option is mutually exclusive with the Host, Port Number, SID, and Service Name options.

Valid Values *path_filename*

where *path_filename* is the entire path, including the file name, to the TNSNAMES.ORA file.

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" on page 427 and "Connection Retry Delay" on page 427 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.

Default None

GUI Tab General tab on page 392

Truststore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 395

Truststore Password

Attribute: TruststorePassword (TSP)

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

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

Valid Values: *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default: None

GUI Tab: Security tab on page 395

Use Current Schema for SQLProcedures

Attribute: UseCurrentSchema (UCS)

Description: Determines whether the driver returns only procedures owned by the current user when executing SQLProcedures.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values: 0 | 1

When set to 1 (Enabled), the call for SQLProcedures is optimized, but only procedures owned by the user are returned.

When set to 0 (Disabled), the driver does not specify only the current user.

Default: 1 (Enabled)

GUI Tab: Performance tab on page 397

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.

You can also use OS Authentication to connect to your Oracle database. See "OS Authentication" on page 473 for details.

Valid Values *userid*

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

Default None

GUI Tab Security tab on page 395

Validate Server Certificate

Attribute ValidateServerCertificate (VSC)

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

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

Valid Values 0 | 1

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must

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

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

Default 1 (Enabled)

GUI Tab Security tab on page 395

Wire Protocol Mode

Attribute WireProtocolMode (WPM)

Description Specifies whether the driver optimizes network traffic to the Oracle server.

This connection option can affect performance. See "Performance Considerations" on page 461 for details.

Valid Values 1 | 2

If set to 1, the driver operates in normal wire protocol mode without optimizing network traffic.

If set to 2, the driver optimizes network traffic to the Oracle server for result sets that contain repeating data in some or all of the columns, and the repeating data is in consecutive rows. It also optimizes network traffic if the application is updating or inserting images, pictures, or long text or binary data.

Default 1

GUI Tab Performance tab on page 397

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: A connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size, if one has been specified. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

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

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

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

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

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

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)

  NOTE: This option is not available with Oracle 8.

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).

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 SQLProcedures (UseCurrentSchema): If your application needs to access database objects owned only by the current user, performance of your application can be improved. In this case, the UseCurrentSchema attribute should be enabled (set to 1). When this attribute 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 attribute is equivalent to passing the Logon ID used on the connection as the SchemaName argument to the catalog functions.

Wire Protocol Mode (WireProtocolMode): Set this option to 2 if:

- Your application executes Select statements that return more than one row, the rows returned have repeating data in some or all of the columns, and the repeated data is in consecutive rows (the data in column1/row1 is the same as the data in column1/row2, for example).
- Your application updates or inserts images, pictures, or long text or binary data.

NOTE: If your application: 1) returns single row result sets or result sets that do not contain repeating data or 2) does not update or insert long data, this option should be set to 1; otherwise, performance may be degraded.

Data Types

Table 9-3 shows how the Oracle data types are mapped to the standard ODBC data types. “Unicode Support” on page 470 lists Oracle to Unicode data type mappings.

Table 9-3. Oracle Data Types

Oracle	ODBC
BFILE [1]	SQL_LONGVARBINARY
BINARY DOUBLE [2]	SQL_REAL
BINARY FLOAT [2]	SQL_DOUBLE
BLOB [2]	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB [2]	SQL_LONGVARCHAR
DATE	SQL_TYPE_TIMESTAMP
LONG	SQL_LONGVARCHAR
LONG RAW	SQL_LONGVARBINARY
NUMBER	SQL_DOUBLE
NUMBER (p,s)	SQL_DECIMAL
RAW	SQL_VARBINARY
TIMESTAMP [3]	SQL_TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE [3]	SQL_TIMESTAMP
TIMESTAMP WITH TIMEZONE [3] [4]	SQL_VARCHAR
VARCHAR2	SQL_VARCHAR
XMLType [5]	SQL_LONGVARCHAR

1. Read-Only
2. Supported only on Oracle 10*g* and higher.
3. Supported only on Oracle 9*i* and higher.
4. Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option only on Oracle 10*g* R2 and higher.
5. Supported only on Oracle 9*i* R2 and higher.

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 will return an Unknown Data Type error (SQL State HY000).

See "Retrieving Data Type Information" on page 77 for more information about data types.

XMLType

Oracle 9*i* R2 and higher supports the XMLType data type. The Oracle Wire Protocol driver supports tables containing columns whose data type is specified as XMLType.

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.

Example

The following example from the Oracle Web site demonstrates how to create a table, insert data, and retrieve data. Creating a table with XMLType is done the same way as creating a regular table in the Oracle database, using standard SQL syntax:

```
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>
'));
```

Use the getClobVal function to retrieve the data:

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

Unicode Support

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

If the database character set is set to UTF-8, the Oracle driver maps the Oracle data types to Unicode data types as shown in the following table:

Oracle Data Type	Mapped to. . .
CHAR	SQL_WCHAR
CLOB	SQL_WLONGVARCHAR
VARCHAR2	SQL_WVARCHAR
LONG	SQL_WLONGVARCHAR

The driver also continues to map these Oracle data types to the normal character data types. See "Data Types" on page 466 for these mappings. The only exception to this is that when the Enable N-CHAR Support option is enabled, the N-CHAR types are mapped to the Unicode types SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR, and the normal character types are mapped to the data types SQL_CHAR, SQL_LONGVARCHAR, and SQL_VARCHAR, regardless of the character set on the Oracle server.

Unexpected Characters

Users are sometimes surprised when they insert a character into a database, only to have a different character displayed when they fetch it from the database. There are many reasons this can happen, but it most often involves code page issues, not driver errors.

Client and server machines in a database system each use code pages, which can be identified by a name or a number, such as Shift_JIS (Japanese) or cp1252 (Windows English). A code page is a mapping that associates a sequence of bits, called a code point, with a specific character. Code pages include the characters and symbols of one or more languages. Regardless of geographical location, a machine can be configured to use a specific code page. Most of the time, a client and database server would use similar, if not identical, code pages. For example, a client and server might use two different Japanese code pages, such as Shift_JIS and EUC_JP, but they would still share many Japanese characters in common. These characters might, however, be represented by different code points in each code page. This introduces the need to convert between code pages to maintain data integrity. In some cases, no one-to-one character correspondence exists between the two code points. This causes a substitution character to be used, which can result in displaying an unexpected character on a fetch.

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage option (see "IANAAppCodePage" on page 441). 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.

MTS Support

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

NOTE: The DataDirect Connect *for* ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 *for* ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

To enable DTC support, you must be connected to an Oracle 8.1.7 or higher server.

OS Authentication

On Windows, UNIX, and Linux, 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.

Support for Oracle RAC

Oracle introduced Real Application Clusters (RAC) with Oracle 9*i*, and RAC is also a key feature of Oracle 10*g*. 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 a DataDirect Connect Series *for* ODBC driver. There is no additional configuration required.

C*onnection 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 "Using Failover" on page 85 for details about how these features work in DataDirect Connect Series *for* ODBC drivers.

Support of Materialized Views

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

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

Stored Procedure Results

When you enable the Procedure Returns Results connection option, the driver returns 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 requires that stored procedures be in a certain format. First, a package must be created to define all of the cursors used in the procedure; then, the procedure can be created using the new cursor. 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.

SSL and Oracle

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 following link explains the supported configurations and how to configure Oracle Enterprise servers for SSL:

http://download.oracle.com/docs/cd/B28359_01/network.111/b28530/asossl.htm#i1013323

SSL negotiates the terms of the encryption in a sequence of events known as the *SSL handshake*. See "Using Security" on page 101 for more information about SSL data encryption.

Using DataDirect Bulk Load on Oracle

DataDirect Bulk Load offers a simple, consistent way to do bulk load operations. See "Using DataDirect Bulk Load" on page 115 for details.

Limitations

The Oracle Wire Protocol driver uses array binding instead of DataDirect Bulk Load in the following situations:

- The Oracle server version is older than Oracle 9*i* R1 (9.0.1)
- External transactions are being used

The Oracle Wire Protocol driver currently does not support the use of LONG and LONG RAW data types with array binding.

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.

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

NOTE: If you are persisting a result set that contains Long data, you must enable the EnableStaticCursorsforLongData connection string attribute.

Isolation and Lock Levels Supported

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

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The Oracle Wire Protocol driver also supports the following functions:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLProcedureColumns
- SQLSetPos
- SQLTablePrivileges

The driver supports the core SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Oracle Wire Protocol driver.

Number of Connections and Statements Supported

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

Using Arrays of Parameters

Oracle 8*i* and higher 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. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

10 The PostgreSQL Wire Protocol Driver

The DataDirect Connect Series *for* ODBC PostgreSQL Wire Protocol driver (the PostgreSQL Wire Protocol driver) is available in both 32- and 64-bit versions and supports the following PostgreSQL database servers:

- PostgreSQL 8.3
- PostgreSQL 8.2

The PostgreSQL Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the Windows, UNIX, and Linux environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the PostgreSQL Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Security
- Connection Pooling

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 494 and Table 10-1 on page 497 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 10-1 on page 497 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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

To configure a PostgreSQL data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

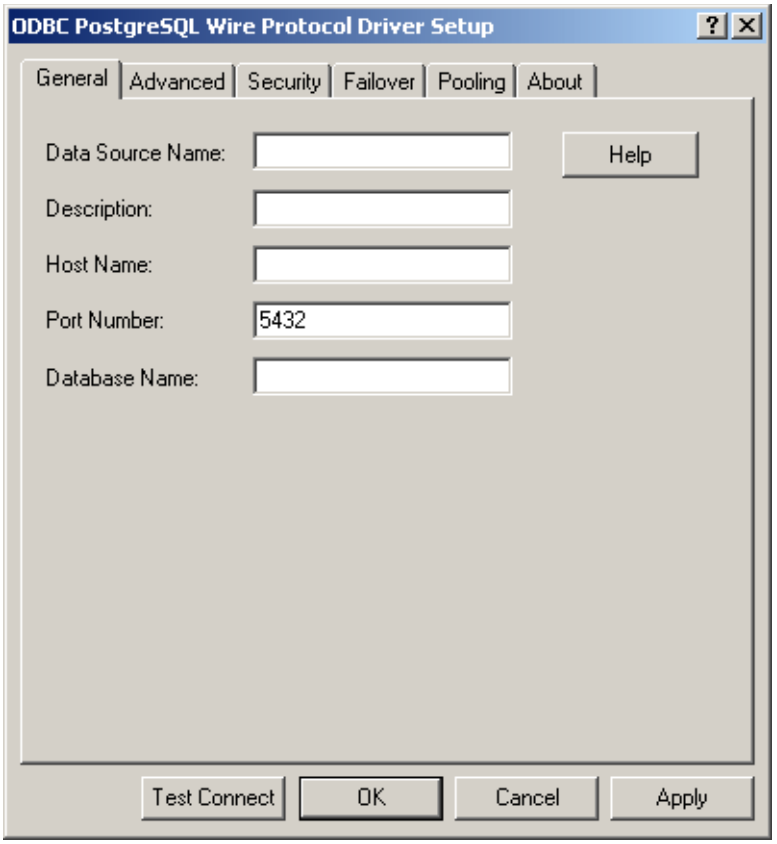
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

The General tab of the Setup dialog box appears.

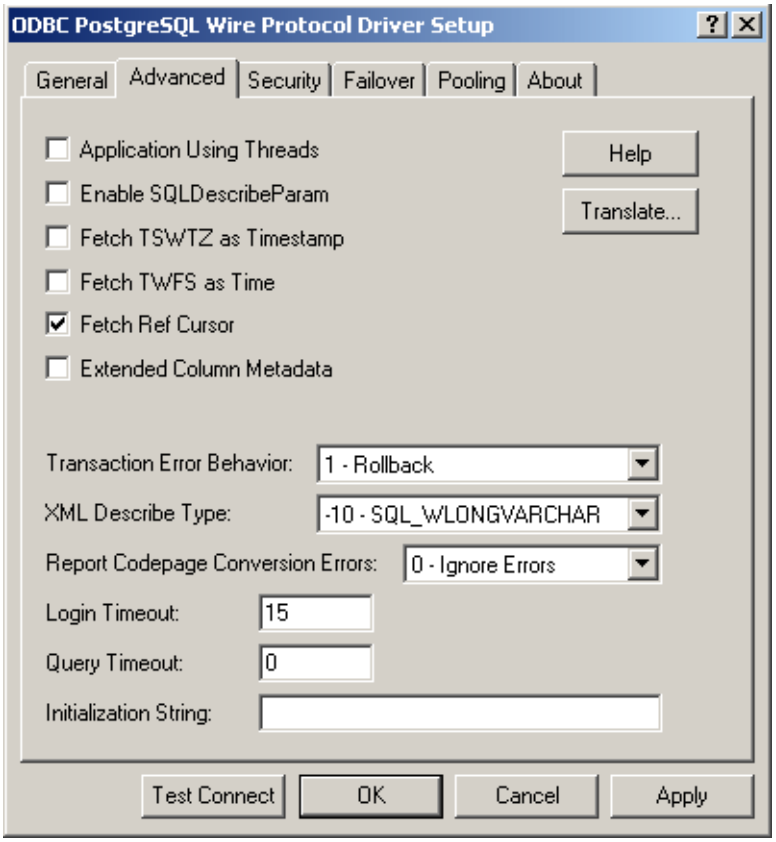


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 502)	None
Description (see page 503)	None
Host Name (see page 510)	None
Port Number (see page 518)	5432
Database Name (see page 503)	None

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



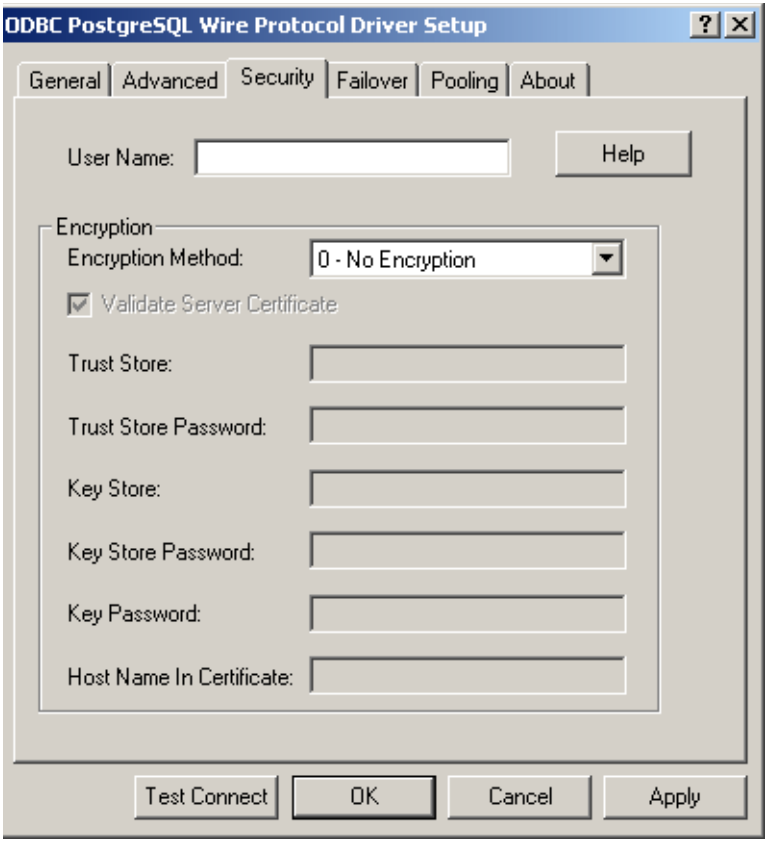
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Using Threads (see page 499)	Enabled
Enable SQLDescribeParam (see page 503)	Disabled
Fetch TSWTZ as Timestamp (see page 509)	Disabled
Fetch TWFS as Time (see page 509)	Disabled
Fetch RefCursors (see page 508)	Enabled
Extended Column Metadata (see page 505)	Disabled
Transaction Error Behavior (see page 520)	1 - Rollback
XML Describe Type (see page 524)	-10 - SQL_WLONGVARCHAR
Report Codepage Conversion Errors (see page 520)	0 - Ignore Errors
Login Timeout (see page 516)	15
Query Timeout (see page 519)	0
Initialization String (see page 513)	None
IANAAppCodePage (see page 512) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

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

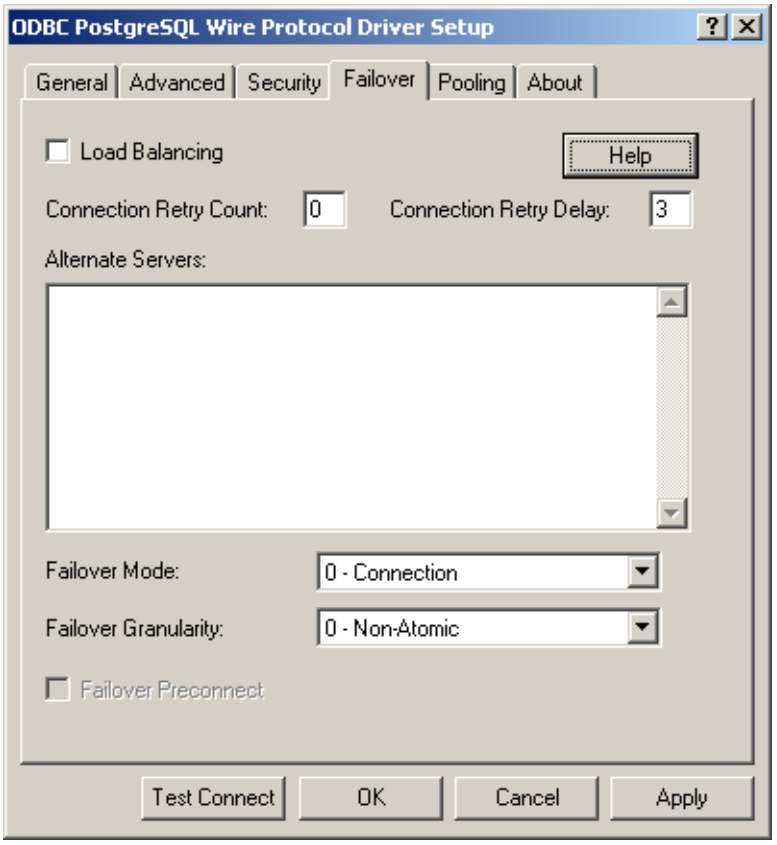


See "Using Security" on page 101 for a general description of encryption and its configuration requirements.

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

Connection Options: Security	Default
User Name (see page 522)	None
Encryption Method (see page 504)	0 - No Encryption
Validate Server Certificate (see page 523)	Enabled
TrustStore (see page 521)	None
Truststore Password (see page 522)	None
Keystore (see page 514)	None
Keystore Password (see page 514)	None
Key Password (see page 513)	None
Host Name In Certificate (see page 511)	None

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



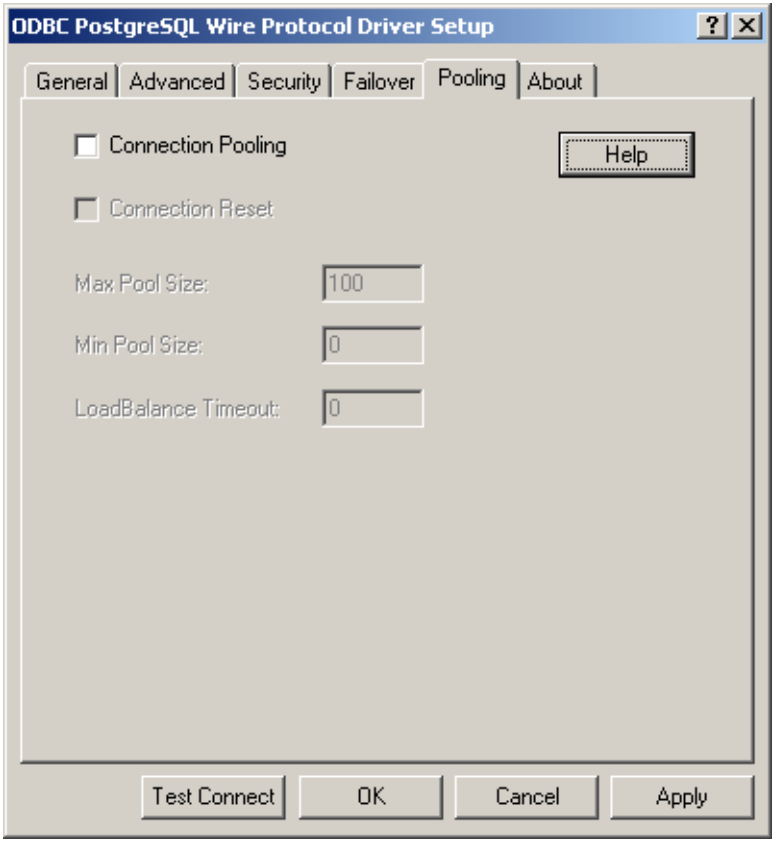
See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 515)	Disabled
Connection Retry Count (see page 501)	0
Connection Retry Delay (see page 502)	3

Connection Options: Failover	Default
Alternate Servers (see page 498)	None
Failover Mode (see page 507)	0 - Connection
Failover Granularity (see page 506)	0 - Non-Atomic
Failover Preconnect (see page 507)	Disabled

7 Optionally, click. the **Pooling** tab to specify pooling data source settings.



See “Using DataDirect Connection Pooling” on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 500)	Disabled
Connection Reset (see page 500)	Disabled
Max Pool Size (see page 517)	100
Min Pool Size (see page 517)	0
Load Balance Timeout (see page 515)	0

8 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 495 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

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

9 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 10-1 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 PostgreSQL Wire Protocol is:

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

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

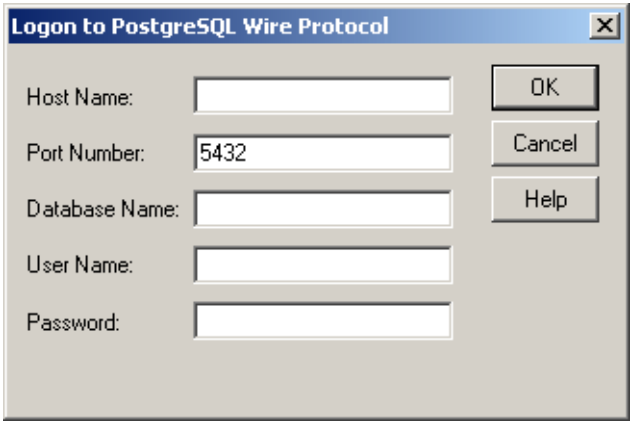
```
FILEDSN=PostgreSQLWP.dsn;UID=JOHN;PWD=XYZZY
```

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

```
DRIVER=DataDirect PostgreSQL Wire Protocol;
HOST=PostgreSQLServer;PORT=5432;UID=JOHN;PWD=XYZZY;DB=
Pgredb1
```

Using a Logon Dialog Box

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



In this dialog box, provide the following information:

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

   The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details concerning these formats.

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

3. In the Database Name field, type the name of the database to which you want to connect.

4. If required, type your PostgreSQL user name.

5. If required, type your PostgreSQL password.

6. Click **OK** to log on to the PostgreSQL database installed on the server you specified and to update the values in the Registry.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an

option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

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

Table 10-1 lists the connection string attributes supported by the PostgreSQL Wire Protocol driver.

Table 10-1. PostgreSQL Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
EnableDescribeParam (EDP)	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
ExtendedColumnMetaData (ECMD)	0 (Disabled)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchRefCursors (FRC)	1 (Enabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Table 10-1. PostgreSQL Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
InitializationString (IS)	None
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
PortNumber (PORT)	5432
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
TransactionErrorBehavior (TEB)	1 (Rollback Transaction)
Truststore (TS)	None
TruststorePassword (TSP)	None
ValidateServerCertificate (VSC)	1 (Enabled)
XML Describe Type (XDT)	-10

Alternate Servers

Attribute AlternateServers (ASVR)

Description A list of alternate database servers to which the driver will try to connect if the primary database server is unavailable. Specifying a value for this option enables connection failover for the driver.

The value you specify must be in the form of a string that defines the physical location of each alternate server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection.

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

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

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

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

```
AlternateServers=(HostName=PostgreSQLServer:
PortNumber=5431:Database=Pgredb1,
HostName=255.201.11.24:PortNumber=5432:Database=Pgredb2)
```

Default None

GUI Tab Failover tab on page 491

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | 1

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

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with

single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI Tab Advanced tab on page 487

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 492

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 492

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 491

Connection Retry Delay

Attribute: ConnectionRetryDelay (CRD)

Description: 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.

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

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

Default: 3

GUI Tab: Failover tab on page 491

Data Source Name

Attribute: DataSourceName (DSN)

Description: 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 on page 487

Database Name

Attribute: Database (DB)

Description: The name of the database to which you want to connect.

Valid Values: *database_name*

where *database_name* is the name of a valid database.

Default: None

GUI Tab: General tab on page 487

Description

Attribute: Description (n/a)

Description: 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 on page 487

Enable SQLDescribeParam

Attribute: EnableDescribeParam (EDP)

Description: Determines whether SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

Valid Values 0 | 1

If set to 1 (enabled), SQLDescribeParam returns the Datatype, ParameterSize, DecimalDigits, and Nullable information for parameters in a prepared statement.

If set to 0 (disabled), the driver does not support SQLDescribeParam and returns the message: `Driver does not support this function`.

Default 0 (Disabled)

GUI Tab Advanced tab on page 487

Encryption Method

Attribute EncryptionMethod (EM)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | 1 | 2

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

Default 0 (No Encryption)

GUI Tab Security tab on page 489

Extended Column Metadata

Attribute ExtendedColumnMetaData (ECMD)

Description Determines how the driver returns column metadata when using SQLDescribCol and SQLColAttribute.

Valid Values 0 | 1

If set to 1 (Enabled), SQLDescribCol returns the actual values for Data Type, Column Size, Decimal Digits, and Nullable. SQLColAttribute returns the actual values for:

- SQL_DESC_CATALOG_NAME
- SQL_DESC_TABLE_NAME
- SQL_DESC_BASE_COLUMN_NAME
- SQL_DESC_LOCAL_TYPE_NAME
- SQL_DESC_NULLABLE
- SQL_DESC_AUTO_UNIQUE_VALUE

If set to 0 (Disabled), SQLDescribCol returns the Data Type, Column Size, and Decimal Digits for the column. The value SQL_NULLABLE_UNKNOWN is returned for Nullable. SQLColAttribute returns the following attribute values:

- SQL_DESC_CATALOG_NAME: empty string
- SQL_DESC_TABLE_NAME: empty string
- SQL_DESC_BASE_COLUMN_NAME: empty string
- SQL_DESC_LOCAL_TYPE_NAME: empty string
- SQL_DESC_NULLABLE: SQL_NULLABLE_UNKNOWN
- SQL_DESC_AUTO_UNIQUE_VALUE: SQL_FALSE

Default 0 (Disabled)

GUI Tab Advanced tab on page 487

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 491

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

Valid Values 0 | 1 | 2

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 on page 491

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

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 on page 491

Fetch RefCursors

Attribute FetchRefCursors (FRC)

Description Determines whether the driver returns refcursors from stored procedures as results sets.

Valid Values 0 | 1

If set to 1 (Enabled), the driver returns refcursors from stored procedures as result sets. The driver fetches all the data from the refcursor and then closes the refcursor. If a stored procedure returns multiple refcursors, the driver generates multiple result sets, one for each refcursor returned.

If set to 0 (Disabled), the driver returns the cursor name for refcursors. The application must fetch the actual data from the refcursor using the cursor name and must close the cursor before additional processing can be done on the statement. The application must close the cursor regardless of whether it actually fetches data from the cursor.

Default 1 (Enabled)

GUI Tab Advanced tab on page 487

Fetch TSWTZ as Timestamp

Attribute FetchTSWTZasTimestamp (FTSWTZAT)

Description Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values 0 | 1

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 timezone 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 timezone information in the fetched value.

Default 0 (Disabled)

GUI Tab Advanced tab on page 487

Fetch TWFS as Time

Attribute FetchTWFSasTime (FTWFSAT)

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

Valid Values 0 | 1

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

If set to 0 (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP.

The fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp

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

Default 0 (Disabled)

GUI Tab Advanced tab on page 487

Host Name

Attribute HostName (HOST)

Description 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" on page 70 for details about these formats.

Default None

GUI Tab General tab on page 487

Host Name In Certificate

Attribute HostNameInCertificate (HNIC)

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

Valid Values *host_name* | #SERVERNAME#

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

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 Common Name parts.

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

Default None

GUI Tab Security tab on page 489

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

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

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

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

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

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 487

Initialization String

Attribute InitializationString (IS)

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

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

Valid Values *SQL_command*

where *SQL_command* is a valid SQL command that is supported by the database.

Example To set the date format on every connection, specify:

```
Set DateStyle='ISO, MDY'
```

Default None

GUI Tab Advanced tab on page 487

Key Password

Attribute KeyPassword (KP)

Description The password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values *key_password*

where *key_password* is the password of a key in the keystore.

Default None

GUI Tab Security tab on page 489

Keystore

Attribute Keystore (KS)

Description The directory of the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file.

Valid Values *keystore_directory*

where *keystore_directory* is the location of the keystore file.

Default None

GUI Tab Security tab on page 489

Keystore Password

Attribute KeystorePassword (KSP)

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

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

Valid Values *keystore_password*

where *keystore_password* is the password of the keystore file.

Default None

GUI Tab Security tab on page 489

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

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

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

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

Default 0

GUI Tab Pooling tab on page 492

Load Balancing

Attribute LoadBalancing (LB)

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

Valid Values 0 | 1

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).

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

Default 0 (Disabled)

GUI Tab Failover tab on page 491

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

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

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 487

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values An integer from 1 to 65535

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

Default 100

GUI Tab Pooling tab on page 492

Min Pool Size

Attribute MinPoolSize (MNPS)

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

This connection option can affect performance. See "Performance Considerations" on page 524 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

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

Default 0

GUI Tab Pooling tab on page 492

Password

Attribute Password (PWD)

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

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

Valid Values *port_name*

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

Default 5432

GUI Tab General tab on page 487

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI Tab Advanced tab on page 487

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 487

Transaction Error Behavior

Attribute TransactionErrorBehavior (TEB)

Description Determines how the driver handles errors that occur within a transaction. When an error occurs in a transaction, the

PostgreSQL server does not allow any operations on the connection except for rolling back the transaction.

Valid Values 0 | 1 | 2

If set to 0 (None), the driver does not roll back the transaction when an error occurs. The application must handle the error and roll back the transaction. Any operation on the statement other than a rollback results in an error.

If set to 1 (Rollback Transaction), the driver rolls back the transaction when an error occurs. In addition to the original error message, the driver posts an error message indicating that the transaction has been rolled back.

If set to 2 (Rollback Savepoint), the driver rolls back the transaction to the last savepoint when an error is detected. In manual commit mode, the driver automatically sets a savepoint after each statement issued. This value makes transaction behavior resemble that of most other database system types, but uses more resources on the database server and may incur a slight performance penalty.

Default 1 (Rollback Transaction)

GUI Tab Advanced tab on page 487

TrustStore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 489

Truststore Password

Attribute TruststorePassword (TSP)

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

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

Valid Values *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default None

GUI Tab Security tab on page 489

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 on page 489

Validate Server Certificate

Attribute ValidateServerCertificate (VSC)

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

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

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Security tab on page 489

XML Describe Type

Attribute: XML Describe Type (XDT)

Description: The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See "Using the XML Data Type" on page 527 for further information about the XML data type.

Valid Values: -4 | -10

If set to -4 (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to -10 (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

-10

GUI Tab: Advanced tab on page 487

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: 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.

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

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

Data Types

Table 10-2 shows how the PostgreSQL data types are mapped to the standard ODBC data types. "Using the XML Data Type" on page 527 describes PostgreSQL to Unicode data type mappings.

Table 10-2. PostgreSQL Data Types

PostgreSQL	ODBC
Bigint	SQL_BIGINT
Bigserial	SQL_BIGINT
Bit [1]	SQL_BIT
Bit varying	SQL_VARBINARY
Boolean	SQL_BIT
Bytea	SQL_VARBINARY
Character	SQL_CHAR
Character varying	SQL_VARCHAR
Date	SQL_TYPE_DATE
Double Precision	SQL_DOUBLE
Integer	SQL_INTEGER
Name	SQL_VARCHAR
Numeric [2]	SQL_NUMERIC
Real	SQL_REAL
Serial	SQL_INTEGER
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR

Table 10-2. PostgreSQL Data Types (cont.)

PostgreSQL	ODBC
Time [3]	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Timestamp with timezone [4]	SQL_VARCHAR
Xml	SQL_LONGVARCHAR

1. Bit maps to SQL_BIT when the length for the bit is 1. If the length is greater than 1, the driver maps the column to SQL_BINARY.
2. Numeric maps to SQL_NUMERIC if the precision of the Numeric is less than or equal to 38. If the precision is greater than 38, the driver maps the column to SQL_VARCHAR.
3. Time mapping changes based on the setting of the Fetch TWFS as Time option
4. Timestamp with timezone mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.

See "Retrieving Data Type Information" on page 77 for more information about data types.

Using the XML Data Type

By default, PostgreSQL returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the option XML Describe Type to SQL_LONGVARBINARY (-4) and bind the data as SQL_C_BINARY.

Unicode Support

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

Stored Procedure Results

PostgreSQL provides functionality to create user-defined functions. PostgreSQL does not make a distinction between user-defined functions and stored procedures. To PostgreSQL, everything is a user-defined function. PostgreSQL does not define a call mechanism for invoking a user-defined function. User-defined functions must be invoked via a SQL statement. For example, given a function defined as:

```
CREATE table foo (intcol int, varcharcol varchar(123))
CREATE or REPLACE FUNCTION insertFoo
  (IN idVal int, IN nameVal varchar) RETURNS void
  AS $$
    insert into foo values ($1, $2);
  $$
  LANGUAGE SQL;
```

Must be invoked natively as:

```
SELECT * FROM insertFoo(100, 'Mark')
```

Even though the function does not return a value or results.

The PostgreSQL Wire Protocol driver supports invoking user-defined functions using the ODBC call Escape. The previously described function can be invoked using:

```
{call insertFoo(100, 'Mark')}
```

PostgreSQL functions return data from functions as a result set. If multiple output parameters are specified, the values for the

output parameters are returned as columns in the result set. For example, the function defined as:

```
CREATE or REPLACE FUNCTION addValues(in v1 int, in v2 int)
  RETURNS int
  AS $$
    SELECT $1 + $2;
  $$
  LANGUAGE SQL;
```

returns a result set with a single column of type SQL_INTEGER, whereas the function defined as:

```
CREATE or REPLACE FUNCTION selectFooRow2
  (IN idVal int, OUT id int, OUT name varchar)
  AS $$
    select id, name from foo where id = $1;
  $$
  LANGUAGE SQL
```

returns a result set that contains two columns, a SQL_INTEGER id column and a SQL_VARCHAR name column.

In addition, when calling PostgreSQL functions that contain output parameters, the native syntax requires that the output parameter values be omitted from the function call. This, in addition to output parameter values being returned as a result set, makes the PostgreSQL behavior of calling functions different from most other databases.

The PostgreSQL Wire Protocol driver provides a mechanism that makes the invoking of functions more consistent with how other databases behave. In particular, the PostgreSQL Wire Protocol driver allows parameter markers for output parameters to be specified in the function argument list when the Escape call is used. The driver allows buffers to be bound to these output parameters. When the function is executed, the output parameters are removed from the argument list sent to the server. The driver extracts the output parameter values from the result set returned by the server and updates the bound output

parameter buffers with those values. For example, the function `selectFooRow2` described previously can be invoked as:

```
sql = L"{call selectFooRow2(?, ?, ?)}";
retVal = SQLPrepare(hPrepStmt, sql, SQL_NTS);
retVal = SQLBindParameter(
  hPrepStmt, 1, SQL_PARAM_INPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf, 0, &idInd);
retVal = SQLBindParameter(
  hPrepStmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
  SQL_INTEGER, 0, 0, &idBuf2, 4, &idInd2);
retVal = SQLBindParameter(
  hPrepStmt, 3, SQL_PARAM_INPUT, SQL_C_WCHAR,
  SQL_VARCHAR, 30, 0, &nameBuf, 123, &nameInd);
retVal = SQLExecute(hPrepStmt);
```

The values of the id and name output parameters are returned in the `idBuf2` and `nameBuf` buffers.

If output parameters are bound to a function call, the driver returns the output parameters in the bound buffers. An error is returned if the number of output parameters bound when the function is executed is less than the number of output parameters defined in the function. If no output parameters are bound to a function call, the driver returns the output parameters as a result set.

PostgreSQL can also return results from a function as a refcursor. There can be, at most, one refcursor per result; however, a function can return multiple results where each result is a refcursor. A connection option defines how the driver handles refcursors. See "Fetch RefCursors" on page 508 for details about this option.

Configuring Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

PostgreSQL supports isolation level 0 (read uncommitted), level 1 (read committed), 2 (Repeatable read), and level 3 (serializable). PostgreSQL supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The PostgreSQL Wire Protocol driver supports the core SQL grammar.

The driver also supports the following functions:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLTablePrivileges

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the PostgreSQL Wire Protocol driver.

Number of Connections and Statements Supported

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

Using Arrays of Parameters

PostgreSQL supports returning a set of output parameters or return values, but no ODBC standard method exists for returning arrays of output parameters or return values. If the call Escape is used to invoke a function that returns a set of output parameters and buffers are bound for those output parameters, the PostgreSQL Wire Protocol driver places the first set of output parameters in the bound buffers. If no output parameters are bound for functions that return a set of results or output parameters, the driver returns a result set with a row for each set of output parameters.

11 The SQL Server Native Wire Protocol Driver

The DataDirect Connect® *for* ODBC SQL Server Native Wire Protocol driver (the SQL Server Native Wire Protocol driver) supports:

- Microsoft SQL Server 2008
- Microsoft SQL Server 2005
- Microsoft SQL Server 2000 Service Packs 1, 2, 3, 3a, and 4
- Microsoft SQL Server 2000 Desktop Engine (MSDE 2000)
- Microsoft SQL Server 2000 Enterprise Edition (64-bit)

The SQL Server Native Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the SQL Server Native Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

The SQL Server Native Wire Protocol driver connects via TCP/IP. TCP/IP connections must be configured on the Windows server on which the Microsoft SQL Server database resides.

Advanced Features

The driver supports the following advanced features:

- Failover
- Connection Retry
- Security
- Connection Pooling
- DataDirect Bulk Load

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Kerberos Authentication Requirements

Verify that your environment meets the requirements listed in Table 11-1 before you configure the driver for Kerberos authentication.

Table 11-1. Kerberos Authentication Requirements for the SQL Server Native Wire Protocol Driver

Component	Requirements
Microsoft SQL Server database server	The database server must be administered by the same domain controller that administers the client and must be running one of the following databases: ■ Microsoft SQL Server 2008 ■ Microsoft SQL Server 2005 ■ Microsoft SQL Server 2000 ■ Microsoft SQL Server 2000 Enterprise Edition (64-bit) Service Pack 2 or higher

Table 11-1. Kerberos Authentication Requirements for the SQL Server Native Wire Protocol Driver *(cont.)*

Component	Requirements
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by Windows Active Directory on one of the following operating systems: ■ Windows Server 2003 ■ Windows 2000 Server Service Pack 3 or higher
Client	The client must be administered by the same domain controller that administers the database server.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 559 and Table 11-2 on page 562 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 11-2 on page 562 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

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

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX

ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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.

Data Source Configuration through a GUI

To configure a Microsoft SQL Server data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

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

2 Select a tab:

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

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

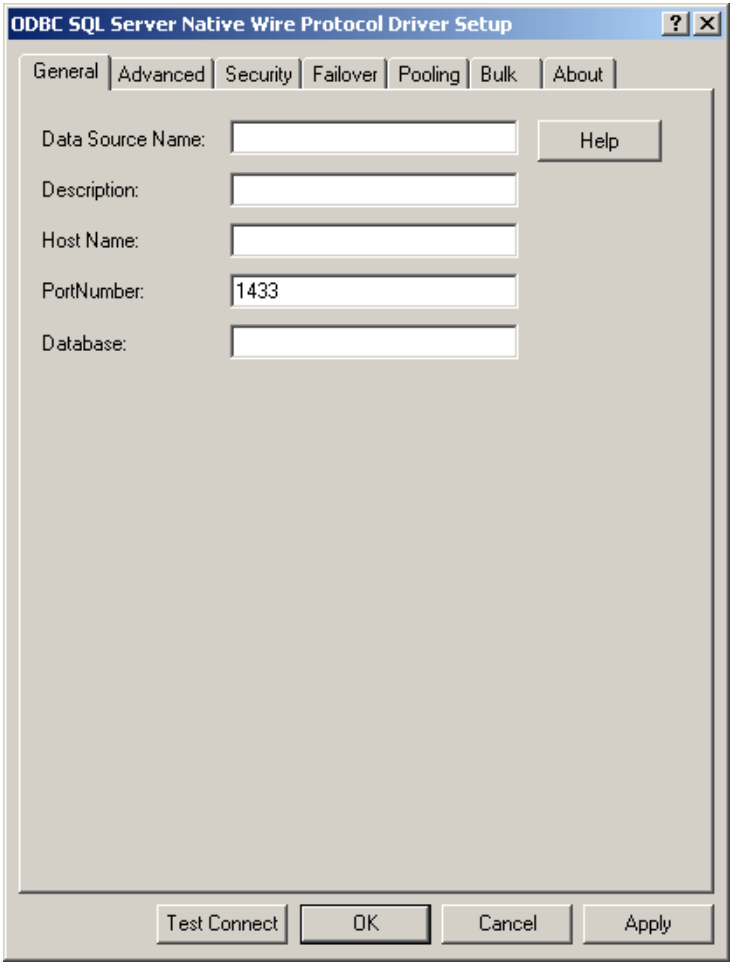
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

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

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

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

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

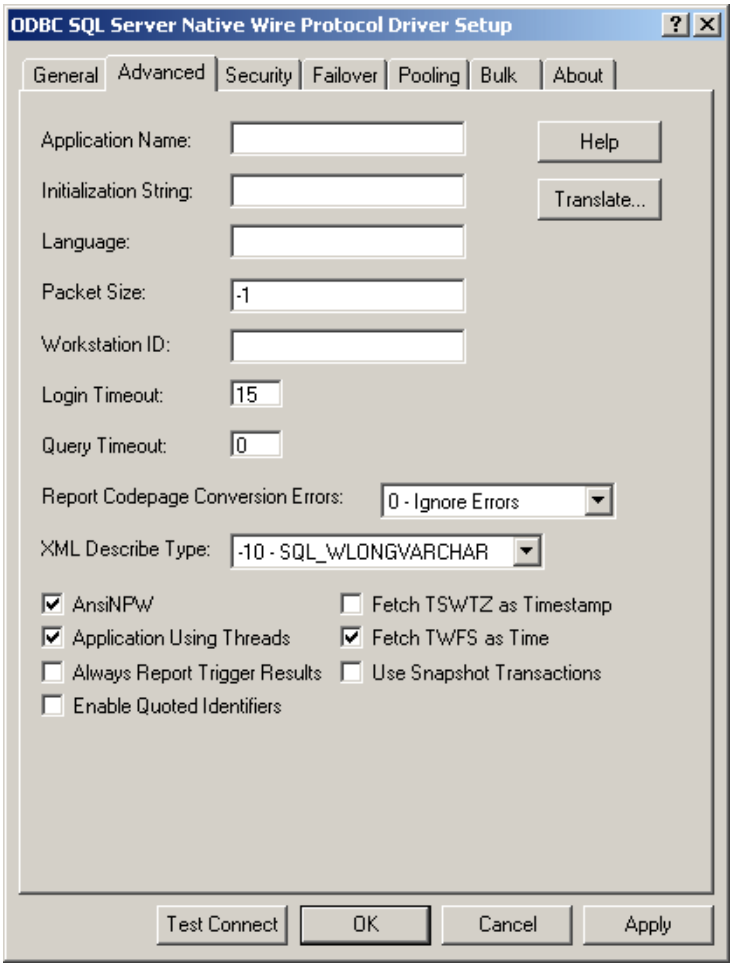


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

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

Connection Options: General	Default
Data Source Name (see page 574)	None
Description (see page 575)	None
Host Name (see page 582)	None
Port Number (see page 591)	1433
Database (see page 574)	None

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



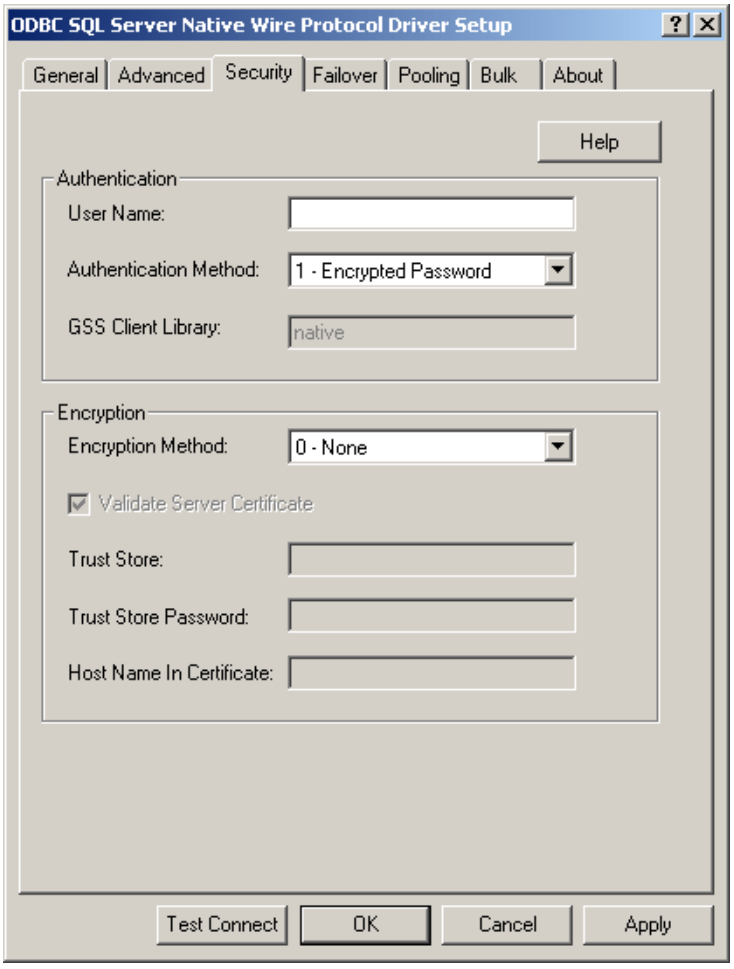
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Name (see page 566)	None
Initialization String (see page 585)	None
Language (see page 585)	None
Packet Size (see page 589)	-1
Workstation ID (see page 596)	None
Login Timeout (see page 587)	15
Query Timeout (see page 591)	0
Report Codepage Conversion Errors (see page 592)	0 (Ignore Errors)
XML Describe Type (see page 597)	-10
AnsiNPW (see page 566)	Enabled
Application Using Threads (see page 567)	Enabled
Always Report Trigger Results (see page 565)	0 (Disabled)
Enable Quoted Identifiers (see page 575)	Disabled
Fetch TWFS as Time (see page 580)	Disabled
Fetch TSWTZ as Timestamp (see page 580)	Disabled
Use Snapshot Transactions (see page 594)	Disabled
IANAAppCodePage (see page 584) UNIX ONLY	4 (ISO 8559-1 Latin-1)

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

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

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

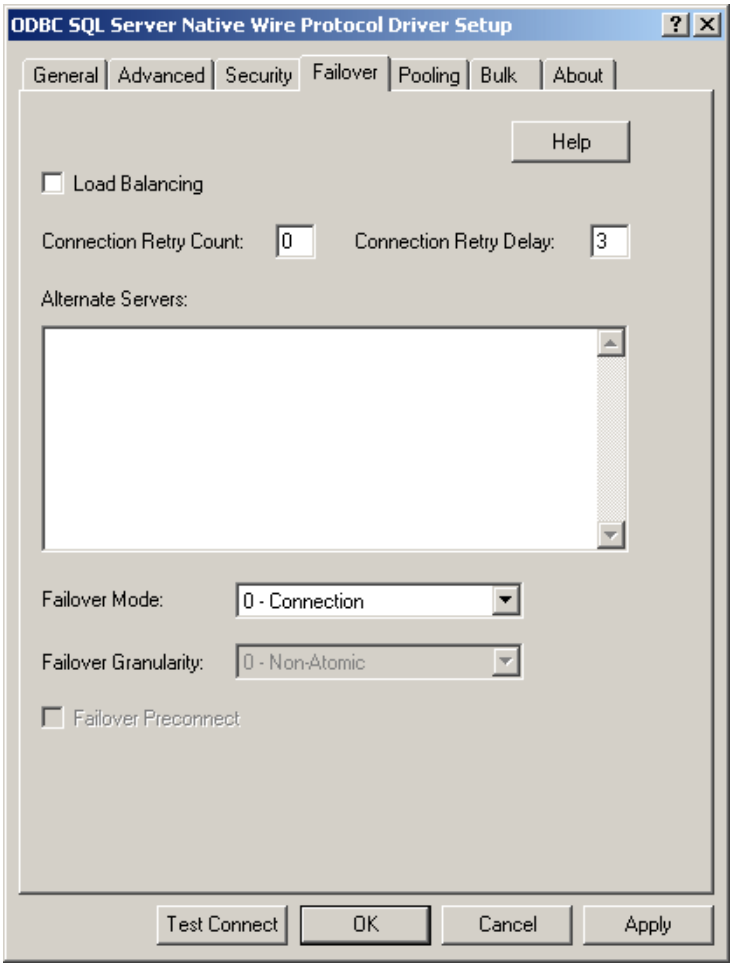


See "Using Security" on page 101 for a general description of authentication and encryption and their configuration requirements.

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

Connection Options: Security	Default
User Name (see page 595)	None
Authentication Method (see page 567)	1 (Encrypt Password)
GSS Client Library (see page 581)	native
Encryption Method (see page 576)	0 (No Encryption)
Validate Server Certificate (see page 595)	Enabled
Truststore (see page 593)	None
Truststore Password (see page 593)	None
Host Name In Certificate (see page 583)	None

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

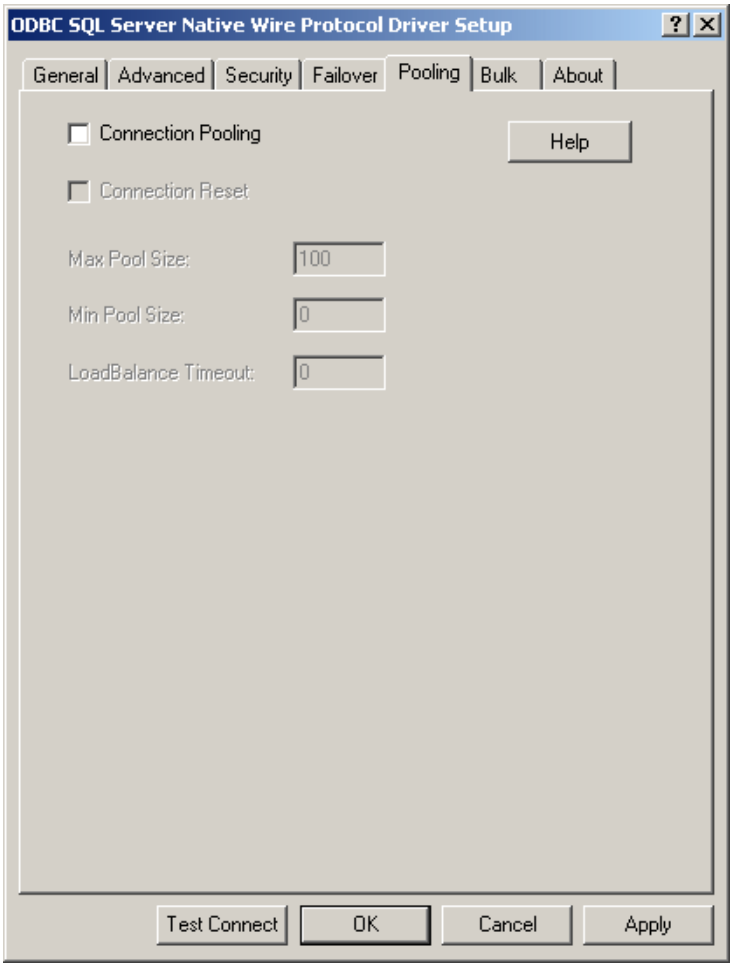


See "Using Failover" on page 85 for a general description of failover and its related connection options.

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

Connection Options: Failover	Default
Load Balancing (see page 586)	Disabled
Connection Retry Count (see page 573)	0
Connection Retry Delay (see page 573)	3
Alternate Servers (see page 564)	None
Failover Mode (see page 578)	0 (Connection)
Failover Granularity (see page 577)	0 (Non-Atomic)
Failover Preconnect (see page 579)	Disabled

7 Optionally, click the **Pooling** tab to specify additional data source settings.

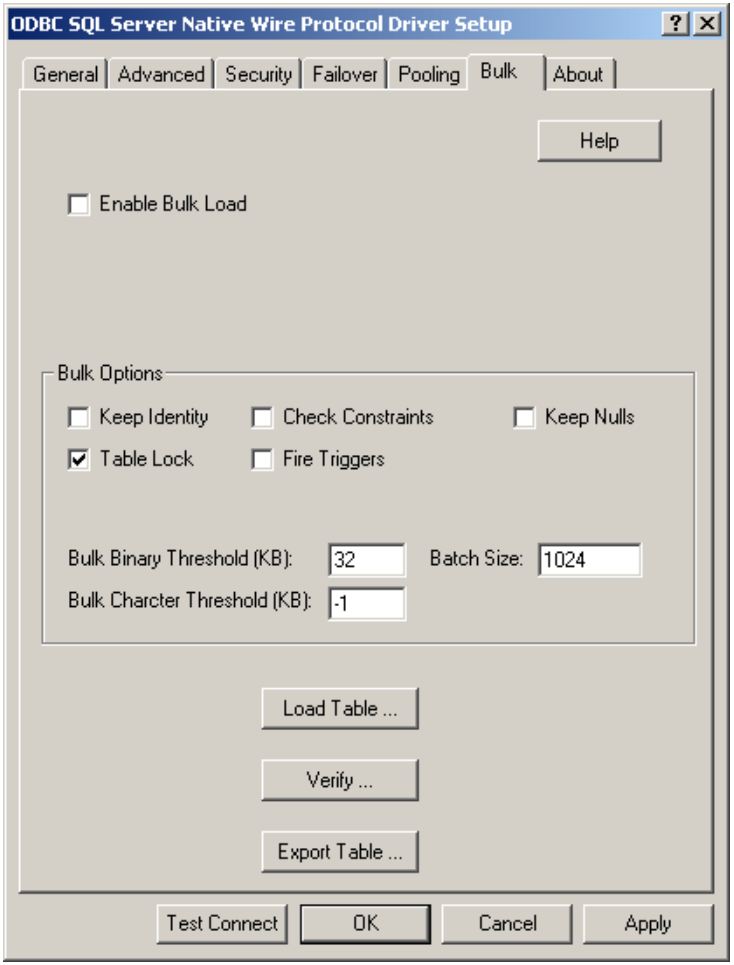


See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

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

Connection Options: Pooling	Default
Connection Pooling (see page 571)	Disabled
Connection Reset (see page 572)	Disabled
Max Pool Size (see page 588)	100
Min Pool Size (see page 588)	0
Load Balance Timeout (see page 586)	0

8 Optionally, click the **Bulk** tab to specify additional data source settings.



See "Using DataDirect Bulk Load" on page 115 for a general description of DataDirect Bulk Load.

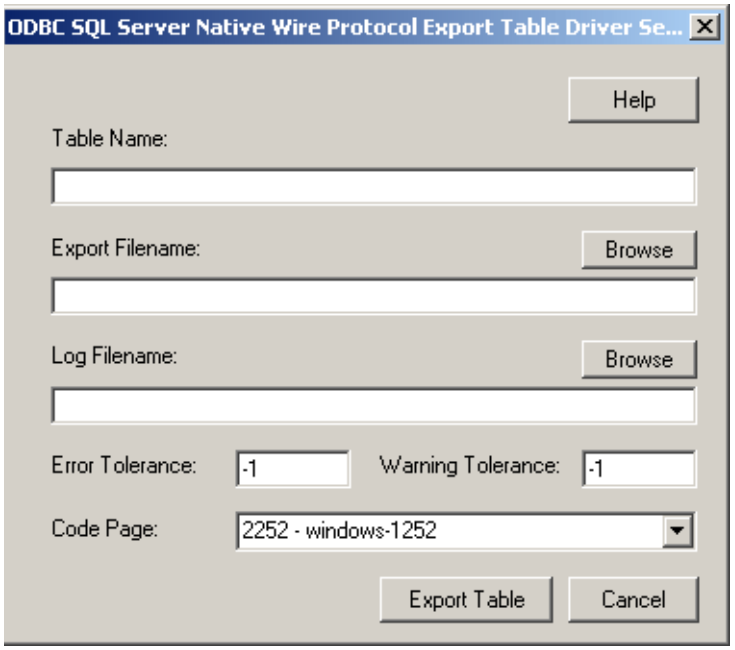
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Bulk	Default
Enable Bulk Load (see page 575)	Disabled
Bulk Options (see page 570) Individual item descriptions are grouped under the Bulk Option description.	Only Table Lock enabled
Bulk Binary Threshold (see page 569)	32
Bulk Character Threshold (see page 569)	-1
Batch Size (see page 568)	1024

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

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

9 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.



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" on page 115 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" on page 129 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. This file must not already exist; if the file already exists, an error is returned.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. Events logged to this file are:

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

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

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

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

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

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

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

Code Page: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See "Character Set Conversions" on page 128 for more information.

The default value on Windows is the current code page of the machine. On UNIX/Linux, the default value is 4.

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

10 To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify** from the Bulk tab. See "Verification of the Bulk Load Configuration File" on page 125 for details. The Verify dialog box appears.

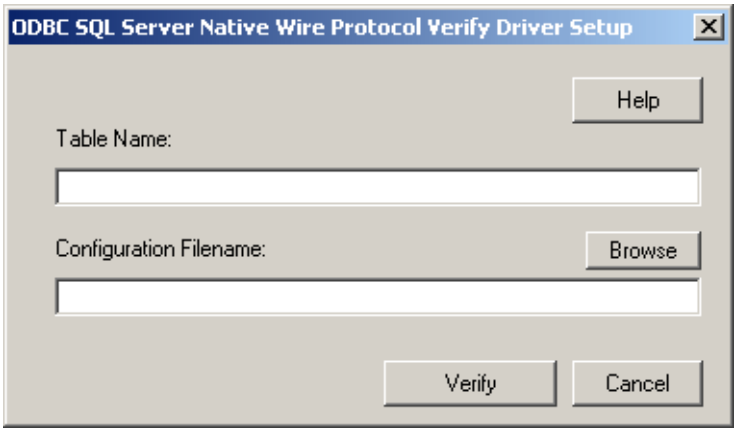
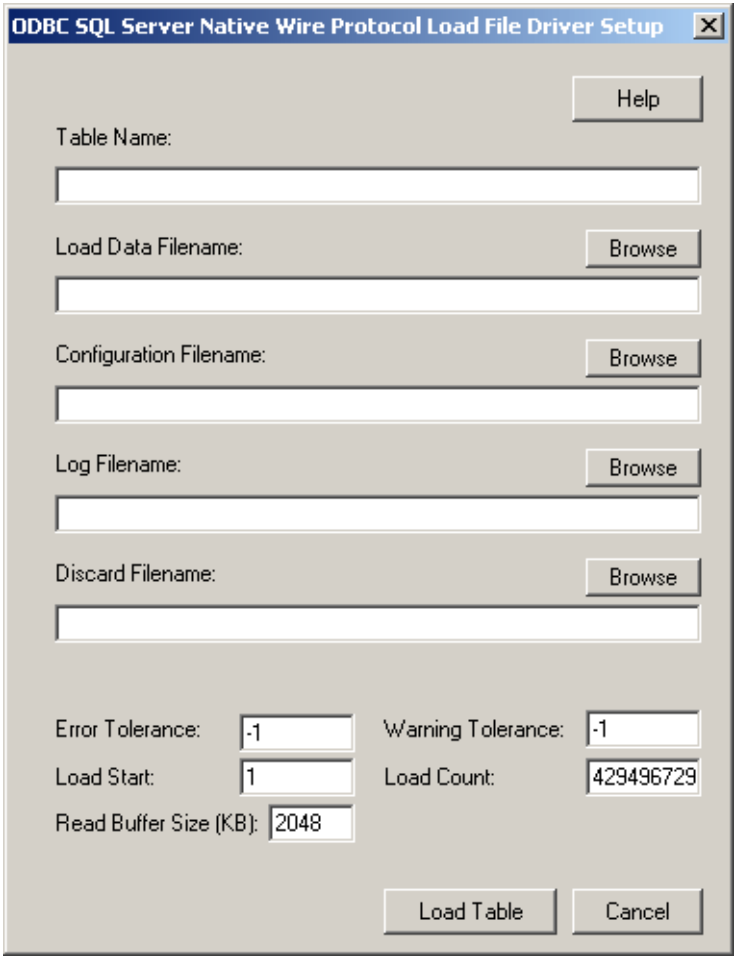


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

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

12 To load data from the bulk data file to a database table, click **Load Table** from the Bulk tab. The Load File dialog box appears.



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 to be 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 to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. Supplying 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 supply 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. 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 you do not supply a value for Discard Filename, no discard 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.

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 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 4294967295.

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

13 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 Connection String" on page 559 for details. Note that the information you enter in the Logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

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

14 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

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

The DSN connection string has the form:

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

The FILEDSN connection string has the form:

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

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

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

Table 11-2 on page 562 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 Microsoft SQL Server is:

```
DSN=ACCOUNTING;DATABASE=ACCT
```

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

```
FILEDSN=SQLServer.dsn;DATABASE=ACCT
```

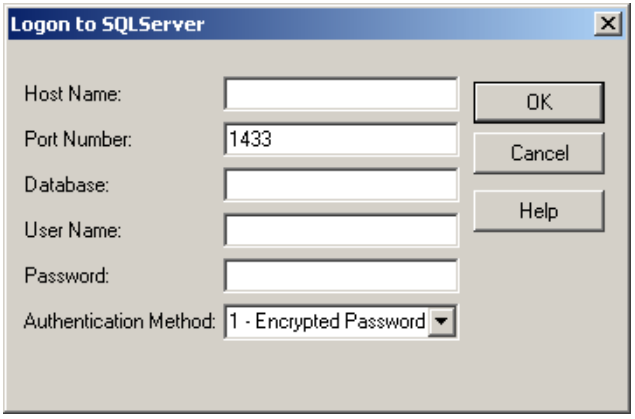
A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect SQL Server Native Wire Protocol;
HOST=SQLServer1;PORT=1433;UID=JOHN;PWD=XYZZY;DB=SQLSdb1
```

Using a Logon dialog Box

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

NOTE: The Login Dialog is not displayed if Authentication Mode has previously been set to Kerberos and the Host Name is specified in the data source.



In the Logon dialog box, provide the following information:

1. Type an IP address in Host Name in following format: *IP_address*. For example, you can enter 199.226.224.34.

   The IP address can be specified in IPv4 on Windows, and in either IPv4 or IPv6 format, or a combination of the two, on UNIX. See "Using IP Addresses" on page 70 for details about these formats.

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

   To specify a named instance of Microsoft SQL Server, use the format: *server_name\instance_name*. If only a server name is specified with no instance name, the driver uses the default instance on the server.

2. Type the Port Number of the server listener.

3. Type the name of the database to which you want to connect. If you do not specify a value, the default database that is defined by Microsoft SQL Server is used.

4. Type your Microsoft SQL Server login ID.

5. Type your password.

6. Select an Authentication Method:

   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 4 - Kerberos, the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

7. Click **OK** to complete the logon and to update the values in the Registry.

Connection Option Descriptions

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

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

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

Table 11-2 lists the connection string attributes supported by the SQL Server Native Wire Protocol driver.

Table 11-2. SQL Server Native Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
Always Report Trigger Results (ARTR)	0 (Disabled)
AnsiNPW (ANPW)	1 (Enabled)
ApplicationName (APP)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
AuthenticationMethod (AM)	1 (Encrypt Password)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1

***Table 11-2. SQL Server Native Wire Protocol Attribute Names** (cont.)*

Attribute (Short Name)	Default
BulkLoadBatchSize (BLBS)	1024
BulkLoadOptions (BLO)	2
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
Database (DB)	None
DataSourceName (DSN)	None
EnableBulkLoad (EBL)	0 (Disabled)
EnableQuotedIdentifiers	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverPreconnect (FP)	0 (Disabled)
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	1 (Enabled)
GSSClientLibrary (GS)	native
HostName (HOST)	None
HostNameInCertificate (HNIC)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
Language (LG)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	None
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
PacketSize (PS)	-1

Table 11-2. SQL Server Native Wire Protocol Attribute Names *(cont.)*

Attribute (Short Name)	Default
Password (PWD)	None
Pooling (POOL)	0 (disabled)
PortNumber (PORT)	1433
QueryTimeout (QT)	0
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
SnapshotSerializable	0 (Disabled)
Truststore (TS)	None
TruststorePassword (TSP)	None
ValidateServerCertificate (VSC)	1 (enabled)
WorkstationID (WSID)	None
XML Describe Type (XDT)	-10

Alternate Servers

Attribute AlternateServers (ASVR)

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

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

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

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

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

```
AlternateServers=(HostName=SqlsServer:
PortNumber=1433:Database=Sqlsdb1,
HostName=255.201.11.24:PortNumber=1434:Database=Sqlsdb2)
```

Default None

GUI Tab Failover tab on page 547

Always Report Trigger Results

Attribute Always Report Trigger Results (ARTR)

Description Determines how the driver reports results that are generated by database triggers (procedures that are stored in the database and executed, or fired, when a table is modified). For Microsoft SQL Server 2005 and higher, this includes triggers that are fired by Data Definition Language (DDL) events.

Valid Values 0 | 1

If set to 1 (Enabled), the driver returns all results, including results that are generated by triggers. Multiple trigger results are returned one at a time. You can use the SQLMoreResults function to return individual trigger results. Warnings and errors are reported in the results as they are encountered.

If set to 0 (Disabled):

- For Microsoft SQL Server 2005 and higher, the driver does not report trigger results if the statement is a single INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, GRANT, REVOKE, or DENY statement.

- For other Microsoft SQL Server databases, the driver does not report trigger results if the statement is a single INSERT, UPDATE, or DELETE statement.

When set to 0, the only result that is returned is the update count that is generated by the statement that was executed (if no errors occurred). Although trigger results are ignored, any errors and warnings that are generated by the trigger are reported. If errors are reported, the update count is not reported.

Default 0 (Disabled)

GUI Tab Advanced tab on page 543

AnsiNPW

Attribute AnsiNPW (ANPW)

Description Determines whether ANSI-defined behaviors are exposed.

Valid Values 0 | 1

When set to 1 (Enabled), the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation. If the driver appears to be truncating trailing blank spaces, this attribute should be set to no.

When set to 0 (Disabled), ANSI-defined behaviors are not exposed.

Default 1 (Enabled)

GUI Tab Advanced tab on page 543

Application Name

Attribute ApplicationName (APP)

Description The name Microsoft SQL Server uses to identify your application.

Valid Values *string*

where *string* is your application name.

Default None

GUI Tab Advanced tab on page 543

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

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

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1

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

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

Default 1 (Enabled)

GUI Tab Advanced tab on page 543

Authentication Method

Attribute AuthenticationMethod (AM)

Description 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 | 4

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

Setting this value to 4 also enables NTLM authentication on Windows platforms.

Default 1 (Encrypt Password)

GUI Tab Security tab on page 545

Batch Size

Attribute BulkLoadBatchSize (BLBS)

Description The number of rows at a time that the driver sends to the database 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.

If set to 0, no rows are sent.

If set to *x*, the specified number of rows are sent.

Default 1024

GUI Tab Bulk tab on page 550

Bulk Binary Threshold

Attribute BulkBinaryThreshold (BBT)

Description 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.

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 on page 550

Bulk Character Threshold

Attribute BulkCharacterThreshold (BCT)

Description 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.

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 on page 550

Bulk Options

Attribute BulkLoadOptions (BLO)

Description Toggles various options for the bulk load process.

Valid Values 0 | *x*

where *x* is a positive integer representing the cumulative total of the values of the bulk load options.

If set to 0, none of the options for bulk load are enabled.

If set to *x*, the values represented by *x* are enabled.

NOTE: The cumulative value of the options is only used in a connection string with the connection string attribute, BulkLoadOptions. On the Advanced tab of the driver Setup dialog, the individual options are enabled by selecting the appropriate check box.

The following bulk load options are available:

- Check Constraints - Checks constraints while data is being inserted. Value=16.

- Fire Triggers - Causes the server to fire the insert triggers for rows being inserted into the database. Value=32.
- Keep Identity - Preserves source identity values. When not enabled, identity values are assigned by the destination. Value=1.
- Keep Nulls - Preserves null values in the destination table regardless of the settings for default values. When not enabled, null values are replaced by column default values, where applicable. Value=64.
- Table Lock - Assigns a table lock for the duration of the bulk copy operation. Other applications are not permitted to update the table during the copy operation. When not enabled, the default bulk locking mechanism (row or table) specified by the table lock on bulk load server option is used. Value=2.

Example If you wanted to enable Check Constraints (16), Fire Triggers (32), and Keep Identity (1) in a connection string, you would add the values together as in the following example:

```
BulkLoadOptions=49
```

To enable these options on the Advanced tab of the driver Setup dialog, you would simply select the check box for each one.

Default 2 (Table Lock enabled)

GUI Tab Bulk tab on page 550

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 549

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 549

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 547

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

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

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

Default 3

GUI Tab Failover tab on page 547

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 542

Database

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 542

Description

Attribute: Description (n/a)

Description: 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 on page 542

Enable Bulk Load

Attribute: EnableBulkLoad (EBL)

Description: Specifies the bulk load method.

Valid Values: 0 | 1

If set to 1 (Enabled), the driver uses the database bulk load protocols. If the protocols cannot be used, the driver returns an error.

If set to 0 (Disabled), the driver uses standard parameter arrays.

Default: 0 (Disabled)

GUI Tab: Bulk tab on page 550

Enable Quoted Identifiers

Attribute: EnableQuotedIdentifiers

Description: Determines whether the driver allows the use of quoted identifiers.

Valid Values 0 | 1

If set to 1 (Enabled), Microsoft SQL Server enforces ANSI rules regarding quotation marks. Double quotation marks can only be used for identifiers, such as column and table names. Character strings must be enclosed in single quotation marks, for example:

```
SELECT "au_id"
FROM "authors"
WHERE "au_lname" = 'O''Brien'
```

If set to no (Disabled), applications that use quoted identifiers encounter errors when they generate SQL statements with quoted identifiers.

Default 0 (Disabled)

GUI Tab Advanced tab on page 543

Encryption Method

Attribute EncryptionMethod (EM)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1 | 6 | 7

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

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

If set to 6 (RequestSSL) - The login request and data are encrypted using SSL if the server is configured for SSL. If the server is not configured for SSL, an unencrypted connection is established.

If set to 7 (LoginSSL) - The login request is encrypted using SSL regardless of whether the server is configured for SSL. The data is encrypted using SSL if the server is configured for SSL, and the data is unencrypted if the server is not configured for SSL.

NOTE: The driver must use the server-specified packet size when using SSL encryption. If SSL is used, any value set for the Packet Size connection option is ignored.

This option can only be set to 1 when Authentication Method is set to 1.

Default 0 (No Encryption)

GUI Tab Security tab on page 545

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 547

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

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

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1 | 2

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 on page 547

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

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 on page 547

Fetch TSWTZ as Timestamp

Attribute FetchTSWTZasTimestamp (FTSWTZAT)

Description Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values 0 | 1

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 timezone 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 timezone information in the fetched value.

Default 0 (Disabled)

GUI Tab Advanced tab on page 543

Fetch TWFS as Time

Attribute FetchTWFSasTime (FTWFSAT)

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

Supported only for Microsoft SQL Server 2008.

Valid Values 0 | 1

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

If set to 0 (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP. The fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp

NOTE: When returning time with fractional seconds data as SQL_TYPE_TIMESTAMP, the Year, Month and Day parts of the timestamp must be set to zero.

Default 1 (Enabled)

GUI Tab Advanced tab on page 543

GSS Client Library

Attribute GSSClientLibrary (GS)

Description The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

The driver uses the standard path for loading the specified client library.

Valid Values native | *client_library*

where *client_library* is a GSS client library installed on the client.

If set to *client_library*, the driver uses the specified GSS client library.

If set to native, the driver uses the GSS client shipped with the operating system.

Default native

GUI Tab Security tab on page 545

Host Name

Attribute HostName (HOST)

Description The name or the IP address of the server to which you want to connect.

Valid Values *IP_address* | *named_server* | *named_instance*

where

IP_address is the IP address of the server to which you want to connect. Specify this address as: *IP_address*. For example, you can enter `199.226.224.34`.

The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details about these formats.

named_server is the named server address of the server to which you want to connect. Specify this address as: *named_server*. For example, you can enter `SSserver`.

named_instance is a named instance of Microsoft SQL Server. Specify this address as: *server_name**instance_name*. If only a server name is specified with no instance name, the driver uses the default instance on the server.

Default None

GUI Tab General tab on page 542

Host Name In Certificate

Attribute HostNameInCertificate (HNIC)

Description A host name for certificate validation when SSL encryption is enabled (Encryption Method=SSL) and validation is enabled (Validate Server Certificate=1). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values *host_name* | #SERVERNAME#

where the *host_name* is the host name specified in the certificate. Consult your SSL administrator for the correct value.

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 Common Name parts.

If set to #SERVERNAME#, then the driver compares the host server name specified as part of a data source or connection string to the dnsName value or the commonName.

Default None

GUI Tab Security tab on page 545

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 543

Initialization String

Attribute InitializationString (IS)

Description A SQL command that is issued immediately after connecting to the database to manage session settings.

NOTE: If the statement fails to execute, the connection fails and the driver reports the error returned from the server.

Valid Values *SQL_command*

where *SQL_command* is a valid SQL command that is supported by the database.

Example To set the date format on every connection, specify:

```
Set DateStyle='ISO, MDY'
```

Default None

GUI Tab Advanced tab on page 543

Language

Attribute Language (LG)

Description The national language to use for Microsoft SQL Server system messages.

Valid Values *lang*

where *lang* is the language to use for Microsoft SQL Server system messages. This overrides the default language specified for the login on the server. If no language is specified, the connection uses the default language specified for the login on the server.

Default None

GUI Tab Advanced tab on page 543

Load Balance Timeout

Attribute LoadBalanceTimeout (LBT)

Description The number of seconds to keep inactive connections open in a connection pool.

NOTE: The Min Pool Size option may cause some connections to ignore this value.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

If set to *x*, inactive connections are closed after the specified number of seconds passes.

Default 0

GUI Tab Pooling tab on page 549

Load Balancing

Attribute LoadBalancing (LB)

Description Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values 0 | 1

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).

NOTE: This option has no effect unless alternate servers are defined for the Alternate Servers connection option.

Default 0 (Disabled)

GUI Tab Failover tab on page 547

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

If set to -1, the connection request does not time out. The driver silently ignores the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 543

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values An integer from 1 to 65535

For example, if set to 20, the maximum number of connections allowed in the pool is 20.

Default 100

GUI Tab Pooling tab on page 549

Min Pool Size

Attribute MinPoolSize (MNPS)

Description The minimum number of connections that are opened and placed in a connection pool, in addition to the active connection, when the pool is created. The connection pool retains this

number of connections, even when some connections exceed their Load Balance Timeout value.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

If set to 0, no connections are opened in addition to the current existing connection.

Default 0

GUI Tab Pooling tab on page 549

Packet Size

Attribute PacketSize (PS)

Description Determines the number of bytes for each database protocol packet that is transferred from the database server to the client machine. Adjusting the packet size can improve performance. The optimal value depends on the typical size of data that is inserted, updated, or returned by the application and the environment in which it is running. Typically, larger packet sizes work better for large amounts of data. For example, if an application regularly returns character values that are 10,000 characters in length, using a value of 32 (16 KB) typically results in improved performance.

NOTE: The ODBC connection option SQL_PACKET_SIZE provides the same functionality as the Packet Size option; however SQL_PACKET SIZE and the Packet Size option are mutually exclusive. If Packet Size is specified, the driver returns the message `Driver Not Capable` if an application attempts to call

SQL_PACKET_SIZE. If you do not set the Packet Size option, application calls to SQL_PACKET_SIZE are accepted by the driver.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values -1 | 0 | *x*

If set to -1, the driver uses the maximum packet size that is set by the database server.

If set to 0, the driver uses the default packet size that is used by the database server.

If set to *x*, an integer from 1 to 127, the driver uses a packet size that is a multiple of 512 bytes. For example, PacketSize=8 means to set the packet size to 8 * 512 bytes (4096 bytes).

NOTE: IF SSL encryption is used, the driver must use the packet size that is specified by the server. Any value set for this option or the SQL_PACKET_SIZE connect option is ignored if SSL encryption is used.

Default -1

GUI Tab Advanced tab on page 543

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

Valid Values *port_name*

where the *port_name* is the port number of the server listener. Check with your database administrator for the correct number.

Default 1433

GUI Tab General tab on page 542

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI Tab Advanced tab on page 543

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 543

Truststore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 545

Truststore Password

Attribute TruststorePassword (TSP)

Description The password that is used to access the truststore file when SSL is enabled (Encryption Method=1) and server authentication is

used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

NOTE: The truststore and keystore files may be the same file; therefore, they may have the same password.

Valid Values *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default None

GUI Tab Security tab on page 545

Use Snapshot Transactions

Attribute SnapshotSerializable

Description Allows your application to use the snapshot isolation level if your Microsoft SQL Server database is configured for Snapshot isolation. Supported only for Microsoft SQL Server 2005 and higher.

See "Using The Snapshot Isolation Level" on page 604 for details about using the snapshot isolation level.

This connection option can affect performance. See "Performance Considerations" on page 597 for details.

Valid Values 0 | 1

When set to 1 (Enabled) and your application has the transaction isolation level set to serializable, the application uses the snapshot isolation level.

When set to 0 (Disabled) and your application has the transaction isolation level set to serializable, the application uses the serializable isolation level.

This option is useful for existing applications that set the isolation level to serializable. Use Snapshot Transactions in this case allows you to change to the snapshot isolation level with no or minimum code changes. If developing a new application, you can code it to set the connection attribute SQL_COPT_SS_TXN_ISOLATION to the value SQL_TXN_SS_SNAPSHOT.

Default 0 (Disabled)

GUI Tab Advanced tab on page 543

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 on page 545

Validate Server Certificate

Attribute ValidateServerCertificate (VSC)

Description Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (Encryption Method=1). When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate returned from the server even if the issuer is not a

trusted CA is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

Truststore information is specified using the Trust Store and Trust Store Password options.

Valid Values 0 | 1

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

If set to 0 (Disabled), the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options.

Default 1 (Enabled)

GUI Tab Security tab on page 545

Workstation ID

Attribute WorkstationID (WSID)

Description The workstation ID that is used by the client.

Valid Values *string*

where *string* is the workstation ID.

Default None

GUI Tab Advanced tab on page 543

XML Describe Type

Attribute: XML Describe Type (XDT)

Description: The SQL data type that is returned by SQLGetTypeInfo for the XML data type.

See "Using the XML Data Type" on page 601 for further information about the XML data type.

Valid Values: -4 | -10

If set to -4 (SQL_LONGVARBINARY), the driver uses the description SQL_LONGVARBINARY for columns that are defined as the XML data type.

If set to -10 (SQL_WLONGVARCHAR), the driver uses the description SQL_WLONGVARCHAR for columns that are defined as the XML data type.

Default

-10

GUI Tab: Advanced tab on page 543

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for details.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: 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.

Enable Bulk Load (EnableBulkLoad): If your application performs bulk loading of data, you can improve performance by configuring the driver to use the database system's bulk load functionality instead of database array binding. The trade-off to consider for improved performance is that using the bulk load functionality can bypass data integrity constraints.

Encryption Method (EncryptionMethod): Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) that is required to encrypt and decrypt data.

Failover Mode (FailoverMode): Although high availability that replays queries after a failure provides increased levels of protection, it can adversely affect performance because of increased overhead.

Packet Size (PacketSize): Typically, it is optimal for the client to use the maximum packet size that the database server allows. This reduces the total number of round trips required to return data to the client, thus improving performance. Therefore, performance can be improved if the PacketSize attribute is set to the maximum packet size of the server.

Use Snapshot Transactions (SnapshotSerializable): You must have your Microsoft SQL Server 2005 and higher database configured for snapshot isolation for this connection option to work. Snapshot Isolation provides transaction-level read consistency and an optimistic approach to data modifications by not acquiring locks on data until data is to be modified. This Microsoft SQL Server 2005 and higher feature can be useful if you want to consistently return the same result set even if another transaction has changed the data and 1) your application executes many read operations or 2) your application has long running transactions that could potentially block users from reading data. This feature has the potential to eliminate data contention between read operations and update operations. When this connection option is enabled, performance is improved due to increased concurrency.

See "Using The Snapshot Isolation Level" on page 604 for details.

Data Types

Table 11-3 shows how the Microsoft SQL Server data types are mapped to the standard ODBC data types. "Unicode Support" on page 602 lists Microsoft SQL Server to Unicode data type mappings.

Table 11-3. Microsoft SQL Server Data Types

SQL Server	ODBC
binary	SQL_BINARY
bigint	SQL_BIGINT
bigint identity	SQL_BIGINT
bit	SQL_BIT
char	SQL_CHAR
date [1]	SQL_TYPE_DATE
datetime	SQL_TYPE_TIMESTAMP
datetime2 [1]	SQL_TYPE_TIMESTAMP
decimal	SQL_DECIMAL
decimal() identity	SQL_DECIMAL
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
int identity	SQL_INTEGER
money	SQL_DECIMAL
numeric	SQL_NUMERIC
numeric() identity	SQL_NUMERIC
real	SQL_REAL
smalldatetime	SQL_TYPE_TIMESTAMP
smallint	SQL_SMALLINT
smallint identity	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sqlvariant	SQL_VARBINARY
text	SQL_LONGVARCHAR
time [1 2]	SQL_TYPE_TIMESTAMP
timestamp	SQL_BINARY
tinyint	SQL_TINYINT
tinyint identity	SQL_TINYINT
uniqueidentifier	SQL_GUID

Table 11-3. Microsoft SQL Server Data Types (cont.)

SQL Server	ODBC
varbinary	SQL_VARBINARY
varbinary(max) [3]	SQL_LONGVARBINARY
varchar	SQL_VARCHAR
varchar(max) [3]	SQL_LONGVARCHAR

1. Supported only on Microsoft SQL Server 2008 and higher.
2. Time mapping changes based on the setting of the Fetch TWFS as Time option.
3. Supported only on Microsoft SQL Server 2005 and higher.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Using the XML Data Type

By default, Microsoft SQL Server returns XML data to the driver encoded as UTF-8. To avoid data loss, an application must bind XML data as SQL_C_WCHAR. The driver then returns the data as either UTF-8 or UTF-16, depending on platform and application settings. If the application binds XML data as SQL_C_CHAR, the driver converts it to the client character encoding, possibly causing data loss or corruption. To prevent any conversion of XML data, the application must set the attribute XML Describe Type to SQL_LONGVARBINARY (-10) and bind the data as SQL_C_BINARY.

Unicode Support

The SQL Server Native Wire Protocol driver maps the Microsoft SQL Server data types to Unicode data types as shown in the following table:

SQL Server Data Type	Mapped to. . .
datetimeoffset [1] [2]	SQL_WVARCHAR
nchar	SQL_WCHAR
ntext	SQL_WLONGVARCHAR
nvarchar	SQL_WVARCHAR
nvarchar(max) [1]	SQL_WLONGVARCHAR
sysname	SQL_WVARCHAR
xml [1]	SQL_WLONGVARCHAR

1. Supported only for Microsoft SQL Server 2008.
2. Datetimeoffset mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Using DataDirect Bulk Load on SQL Server

DataDirect Bulk Load offers a simple, consistent way to do bulk load operations. See "Using DataDirect Bulk Load" on page 115 for details.

For optimal performance, you must enable minimal logging and Table Locking. Please refer to the following Web sites for more information on minimal logging:

http://msdn.microsoft.com/en-us/library/ms190422.aspx

http://msdn.microsoft.com/en-us/library/ms190203.aspx

Table Locking, one of the Bulk Options, is enabled by default. This prevents other transactions from accessing the table during bulk load. See "Bulk Options" on page 570 for details about this option.

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

Microsoft SQL Server supports isolation levels 0 (Read Uncommitted), 1 (Read Committed), 2 (Repeatable Read), and 3 (Serializable). Microsoft SQL Server supports row-level and table-level locking. Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

Microsoft SQL Server 2005 and higher supports the following additional isolation levels:

- Snapshot
- Read Committed with Snapshots
- Read Committed with Locks (equivalent to Read Committed in previous Microsoft SQL Server versions)

Using The Snapshot Isolation Level

The Snapshot isolation level is available only with Microsoft SQL Server 2005 and higher. Setting the SnapshotSerializable connection string attribute changes the behavior of the Serializable isolation level to use the Snapshot Isolation level. This allows an application to use the Snapshot Isolation level with minimal or no code changes.

If you are writing a new application, you may want to code it to set the connection attribute SQL_COPT_SS_TXN_ISOLATION to the value SQL_TXN_SS_SNAPSHOT. The application then uses the snapshot isolation level without requiring the Use Snapshot Transactions connection option.

See "Use Snapshot Transactions" on page 594 for additional information.

ODBC Conformance Level

The SQL Server Native Wire Protocol driver supports ODBC conformance level 1.

In addition, the following level 2 functions are supported:

- SQLForeignKeys
- SQLTablePrivileges
- SQLDescribeParam
- SQLColumnPrivileges

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the SQL Server Native Wire Protocol driver.

Number of Connections and Statements Supported

The SQL Server Native Wire Protocol driver supports multiple connections and multiple statements per connection.

Using Arrays of Parameters

Microsoft SQL Server databases natively support parameter arrays, and the SQL Server Native 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. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

12 The Sybase Wire Protocol Driver

The DataDirect Connect Series *for* ODBC Sybase Wire Protocol driver (the Sybase Wire Protocol driver) is available in both 32- and 64-bit versions and supports the following Sybase database servers:

- Sybase Adaptive Server Enterprise 15
- Sybase Adaptive Server Enterprise 12.0 and 12.5.*x*
- Sybase Adaptive Server 11.5 and higher

The Sybase Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the Sybase Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry
- Security
- Connection Pooling
- DataDirect Bulk Load

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Kerberos Authentication Requirements

Verify that your environment meets the requirements listed in Table 12-1 before you using the driver for Kerberos authentication.

Table 12-1. Kerberos Authentication Requirements for the Sybase Wire Protocol Driver

Component	Requirements
Database server	The database server must be administered by the same domain controller that administers the client and must be running Sybase 12.0 or higher. In addition, the Sybase Security and directory services package, ASE_SECDIR, is required.

Table 12-1. Kerberos Authentication Requirements for the Sybase Wire Protocol Driver *(cont.)*

Component	Requirements
Kerberos server	The Kerberos server is the machine where the user IDs for authentication are administered. The Kerberos server is also the location of the Kerberos KDC. Network authentication must be provided by one of the following methods: ■ Windows Active Directory on one of the following operating systems: Windows Server 2003 or Windows 2000 Server Service Pack 3 or higher ■ MIT Kerberos 1.4.2 or higher
Client	The client must be administered by the same domain controller that administers the database server.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 635 and Table 12-2 on page 639 for an alphabetical list of driver connection string attributes and their initial default values

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 12-2 on page 639 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Sybase data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

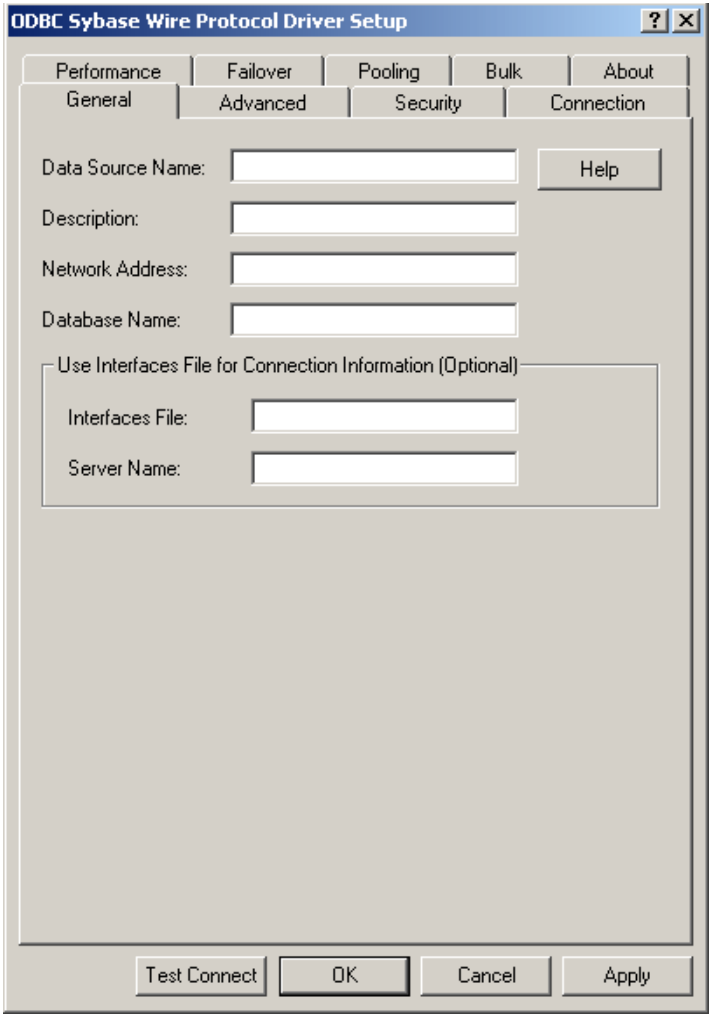
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

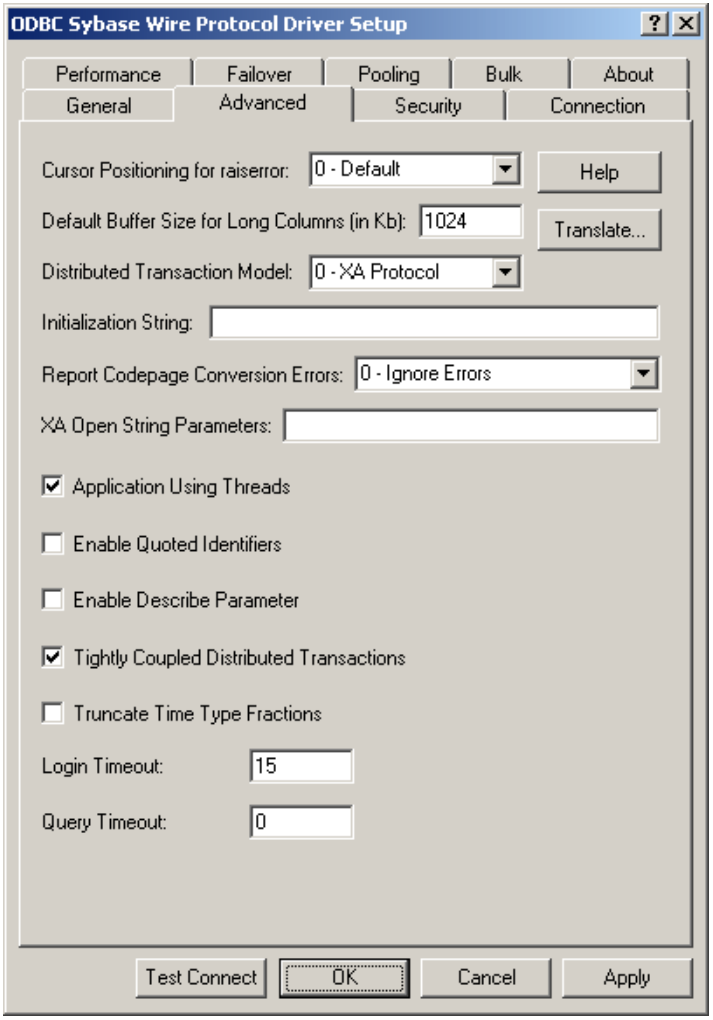


NOTE: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 650)	None
Description (see page 652)	None
Network Address (see page 669)	None
Database Name (see page 651)	None
Interfaces File (see page 663)	None
Server Name (see page 675)	None

4 Optionally, click the **Advanced** tab to specify additional data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

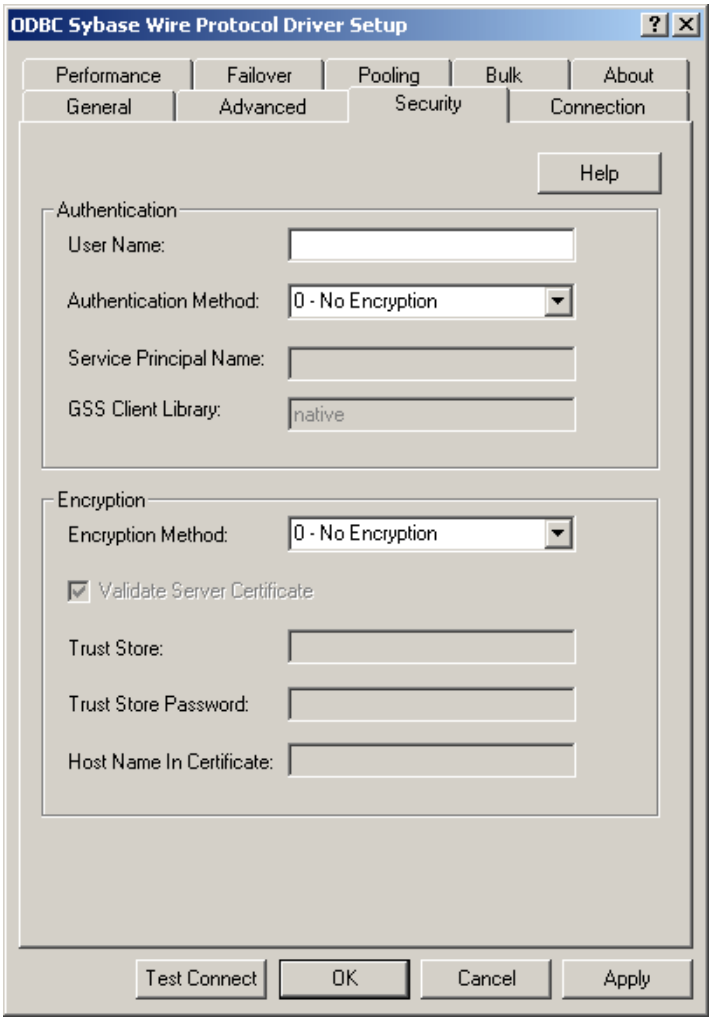
Connection Options: Advanced	Default
Cursor Positioning for Raiserror (see page 650)	0 - Default

Connection Options: Advanced	Default
Default Buffer Size for Long/LOB Columns (in Kb) (see page 652)	1024
Distributed Transaction Model (see page 653)	0 - XA Protocol
Initialization String (see page 662)	None
Report Codepage Conversion Errors (see page 673)	0 - Ignore Errors
XA Open String Parameters (see page 679)	None
Application Using Threads (see page 642)	Enabled
Enable Quoted Identifiers (see page 654)	Disabled
Enable Describe Parameter (see page 654)	Disabled
Tightly Coupled Distributed Transactions (see page 676)	Enabled
Truncate Time Type Fractions (see page 676)	Disabled
Login Timeout (see page 667)	15
Query Timeout (see page 672)	0
IANAAppCodePage (see page 661) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

5 Optionally, click the **Security** tab to specify security data source settings.

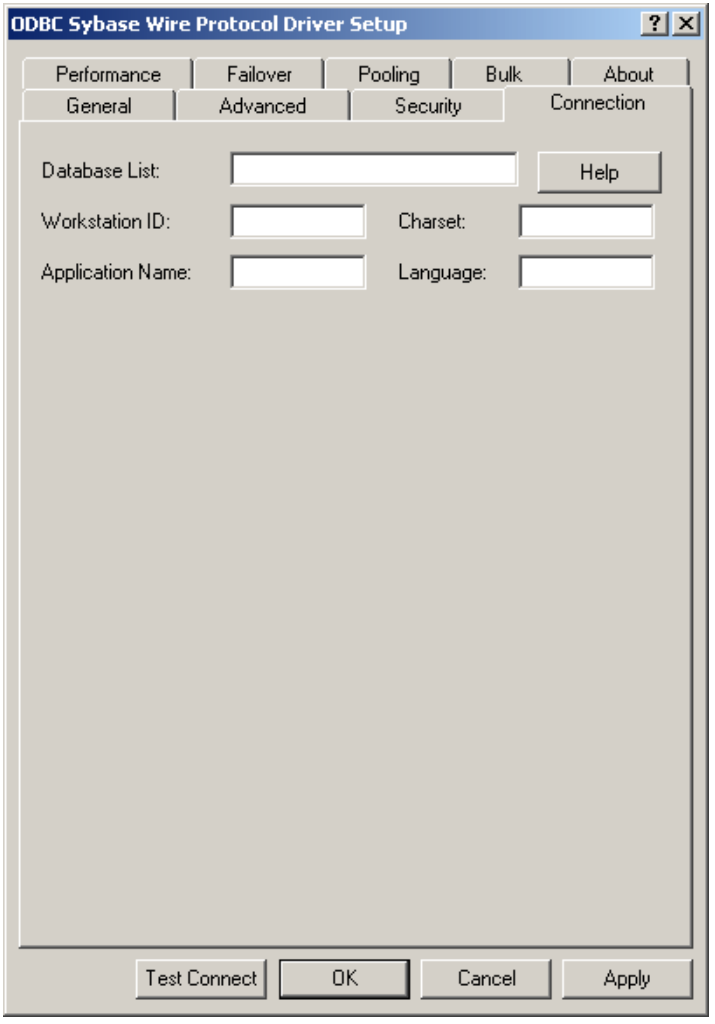


On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Security	Default
User Name (see page 678)	None
Authentication Method (see page 643)	0 - No Encryption
Service Principal Name (see page 675)	None
GSS Client Library (see page 658)	native
Encryption Method (see page 655)	0 - No Encryption
Validate Server Certificate (see page 678)	Enabled
Truststore (see page 677)	None
Truststore Password (see page 677)	None
Host Name In Certificate (see page 660)	None

See "Using Security" on page 101 for a general description of authentication and encryption and their configuration requirements.

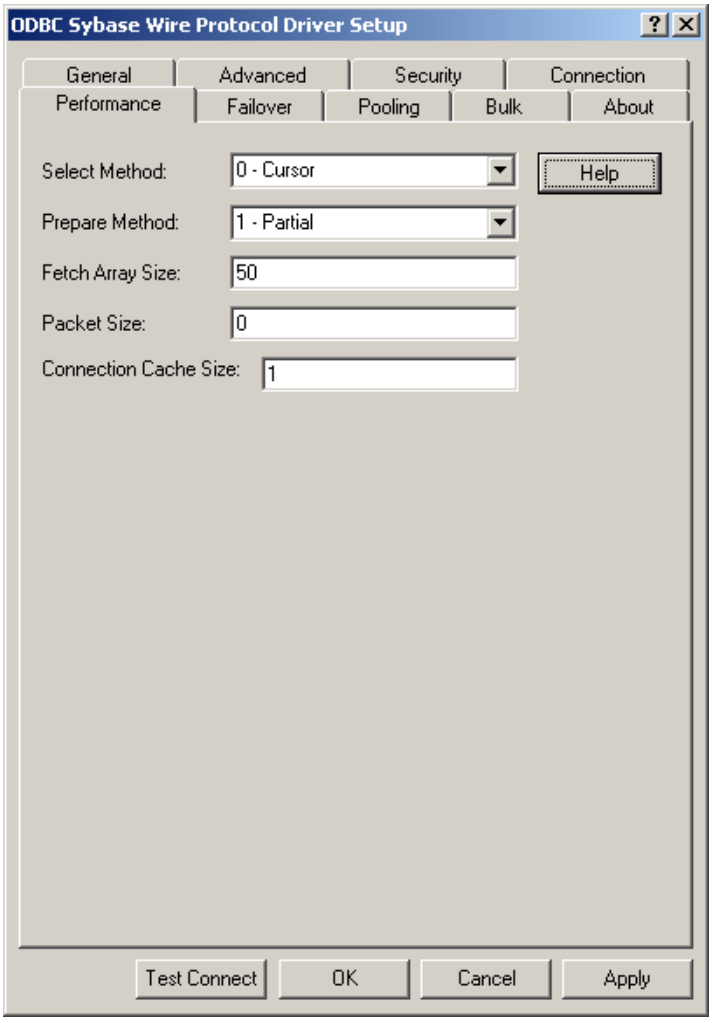
6 Optionally, click the **Connection** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Connection	Default
Database List (see page 651)	None
Workstation ID (see page 679)	None
Charset (see page 646)	None
Application Name (see page 642)	None
Language (see page 665)	None

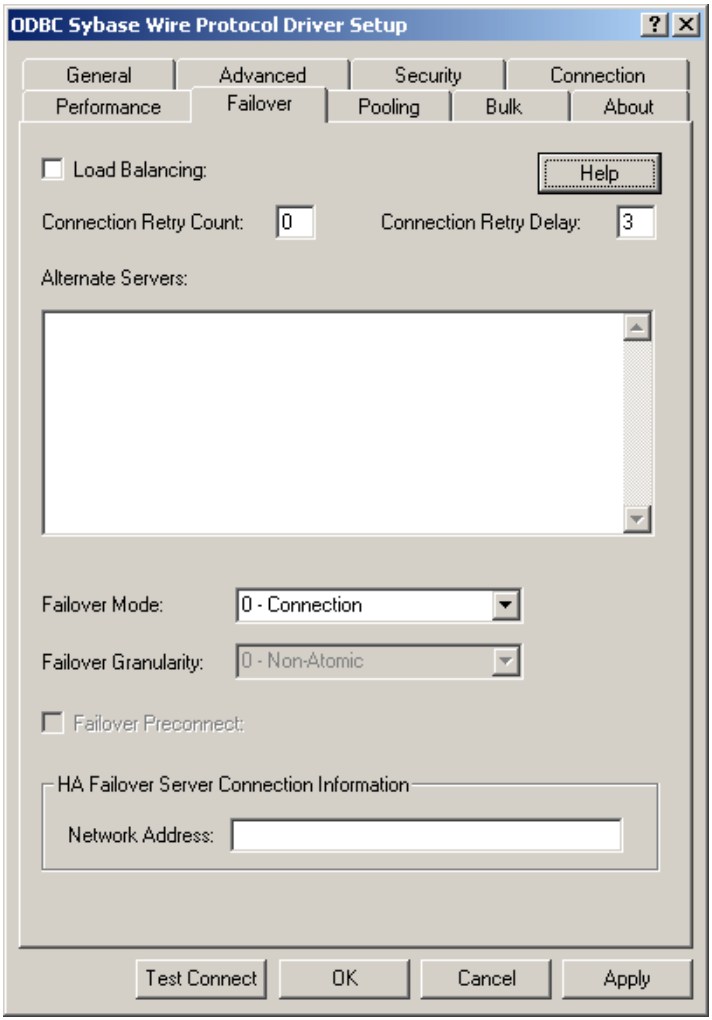
7 Optionally, click the **Performance** tab to specify performance data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Performance	Default
Select Method (see page 674)	0 - Cursor
Prepare Method (see page 671)	1 - Partial
Fetch Array Size (see page 658)	50
Packet Size (see page 670)	0
Connection Cache Size (see page 647)	1

8 Optionally, click the **Failover** tab to specify failover data source settings.

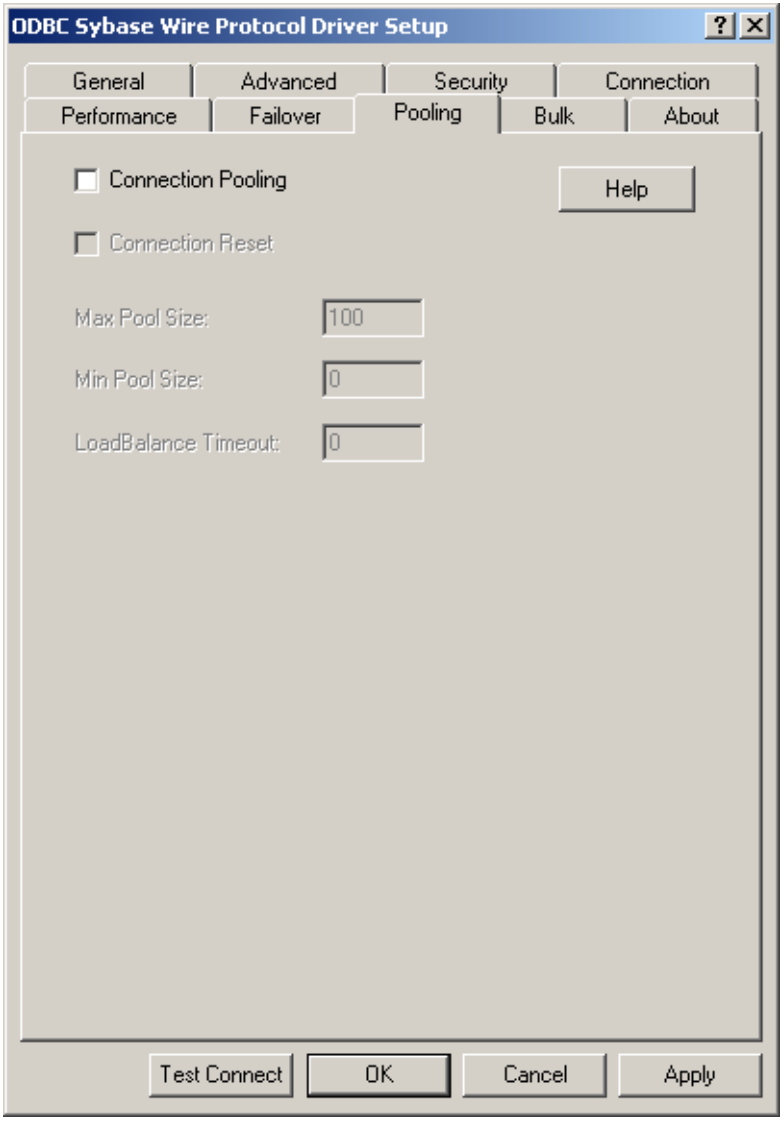


See "Using Failover" on page 85 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing (see page 666)	Disabled
Connection Retry Count (see page 649)	0
Connection Retry Delay (see page 649)	3
Alternate Servers (see page 641)	None
Failover Mode (see page 656)	0 - Connection
Failover Granularity (see page 655)	0 - Non-Atomic
Failover Preconnect (see page 657)	Disabled
HA Failover Server Connection Information/Network Address (see page 659)	None

9 Optionally, click the **Pooling** tab to specify connection pooling data source settings.

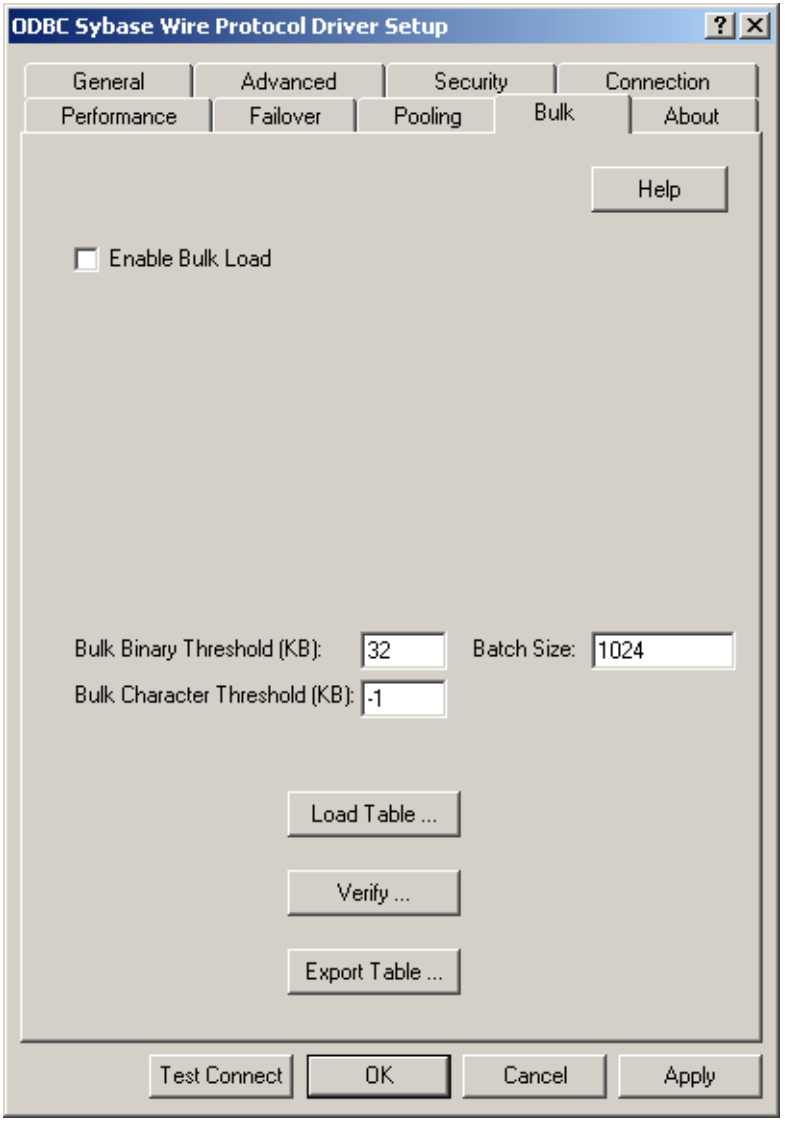


See "Using DataDirect Connection Pooling" on page 109 for a general description of connection pooling.

On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Pooling	Default
Connection Pooling (see page 647)	Disabled
Connection Reset (see page 648)	Disabled
Max Pool Size (see page 668)	100
Min Pool Size (see page 668)	0
Load Balance Timeout (see page 666)	0

10 Optionally, click the **Bulk** tab to specify DataDirect Bulk Load data source settings.



See "Using DataDirect Bulk Load" on page 115 for a general description of DataDirect Bulk Load.

On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Bulk	Default
Enable Bulk Load (see page 653)	Disabled
Bulk Binary Threshold (see page 644)	32
Bulk Character Threshold (see page 645)	-1
Batch Size (see page 644)	1024

If your application is already coded to use parameter array batch functionality, you can leverage DataDirect Bulk Load features through the Enable Bulk Load connection option. Enabling this option automatically converts the parameter array batch operation to use the database bulk load protocol.

If you are not using parameter array batch functionality, you can export data to a bulk load data file, verify the metadata of the bulk load configuration file against the structure of the target table, and bulk load data to a table. Use the following steps to accomplish these tasks.

11 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.

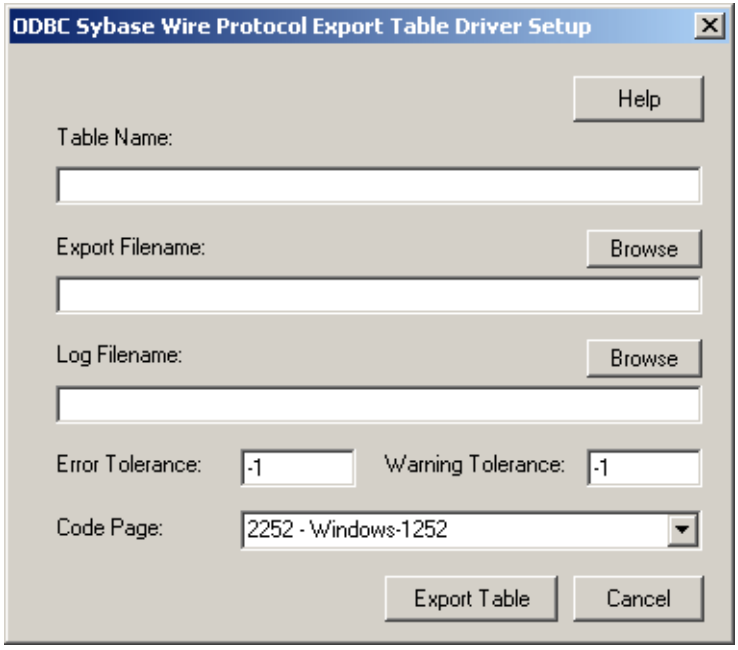


Table Name: A string that specifies the name of the source database table containing the data to be exported.

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. This file must not already exist; if the file already exists, an error is returned.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. The log file is created if it does not exist. Events logged to this file are:

- Total number of rows fetched
- A message for each row that failed to export
- Total number of rows that failed to export
- Total number of rows successfully exported

Information about the load is written to this file, preceded by a header. Information about the next load is appended to the end of the file.

If you do not supply a value for Log Filename, no log file is created.

Error Tolerance: A value that specifies the number of errors to tolerate before an operation terminates. A value of 0 indicates that no errors are tolerated; the operation fails when the first error is encountered.

The default of -1 means that an infinite number of errors is tolerated.

Warning Tolerance: A value that specifies the number of warnings to tolerate before an operation terminates. A value of 0 indicates that no warnings are tolerated; the operation fails when the first warning is encountered.

The default of -1 means that an infinite number of warnings is tolerated.

Code Page: A value that specifies the code page value to which the driver must convert all data for storage in the bulk data file. See "Character Set Conversions" on page 128 for more information.

The default value on Windows is the current code page of the machine. On UNIX/Linux, the default value is 4.

12 Click **Export Table** to connect to the database and export data to the bulk data file or click **Cancel**.

13 To verify the metadata of the bulk load configuration file against the structure of the target database table, click **Verify** from the Bulk tab. See "Verification of the Bulk Load Configuration File" on page 125 for details. The Verify dialog box appears.

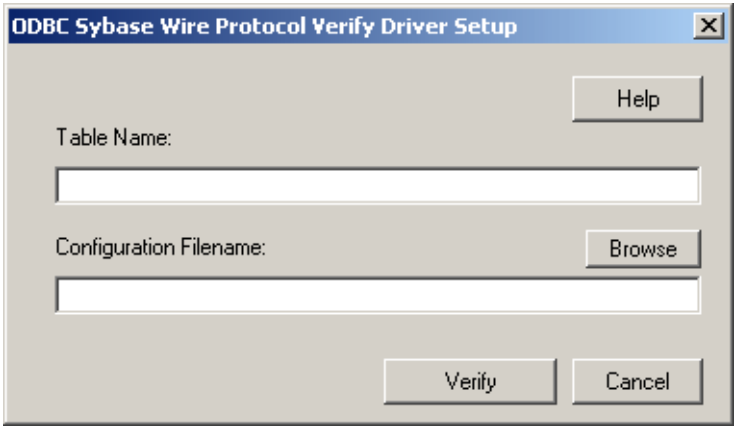


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

14 Click **Verify** to verify table structure or click **Cancel**.

15 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.

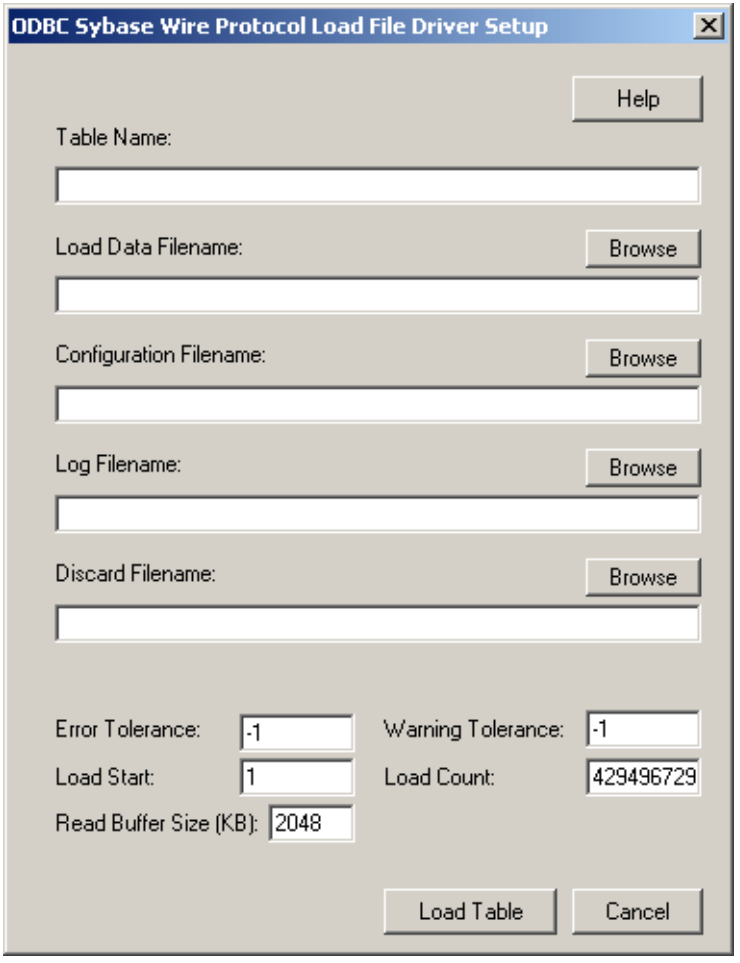


Table Name: A string that specifies the name of the target database table into which the data is to be loaded.

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 to be loaded.

Configuration Filename: A string that specifies the path (relative or absolute) and file name of the bulk configuration file.

Log Filename: A string that specifies the path (relative or absolute) and file name of the bulk log file. Supplying 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 supply 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. 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 you do not supply a value for Discard Filename, no discard 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.

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 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 4294967295.

16 Click **Load Table** to connect to the database or click **Cancel**.

17 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" on page 637 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

NOTE: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

18 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

`DSN=data_source_name[;attribute=value[;attribute=value]...]`

The FILEDSN connection string has the form:

`FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]`

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 12-2 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Sybase is:

```
DSN=SYB TABLES;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

A FILEDSN connection string is similar except for the initial keyword:

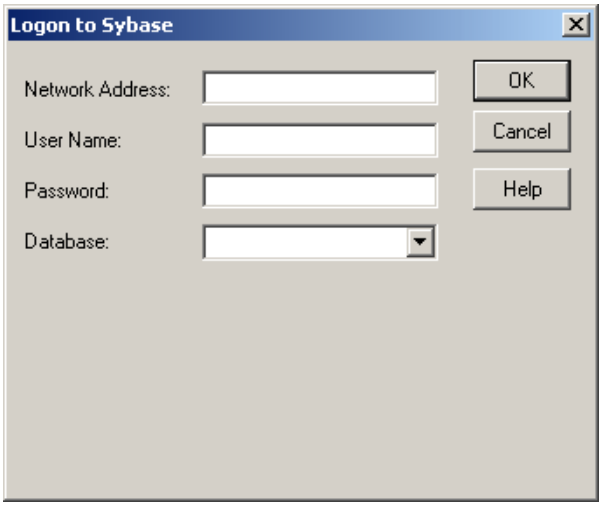
```
FILEDSN=SYB.dsn;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect Sybase Wire Protocol;
NA=123.456.78.90, 5000;NLN=Winsock;
DB=SYBACCT;UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box

Some ODBC applications display a Logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.



In the Logon dialog box, provide the following information:

1. In the Network Address field, specify an IP address for the Sybase server as follows: *IP address*,*port_number*. For example, you might enter `199.226.224.34,5000`. If your network supports named servers, you can specify an address as: *servername*,*port_number*. For example, you might enter `Sybaseserver,5000`.

   The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details concerning these formats.

2. If required, type your case-sensitive login ID.

3. If required, type your case-sensitive password for the system.

4 In the Database field, type the name of the database you want to access (case-sensitive). Or, select the name from the Database drop-down list, which displays the names that you specified on the Connection tab of the ODBC Sybase Wire Protocol driver Setup dialog box.

   NOTE: If you are connecting through the **Test Connect** button of the Setup dialog box, only the default database specified on the General tab of the Setup dialog box is available in the Database drop-down list. The database names specified on the Connection tab are not available.

5 Click **OK** to complete the logon and to update the values in the Registry.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 12-2 lists the connection string attributes supported by the Sybase Wire Protocol driver.

Table 12-2. Sybase Wire Protocol Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationName (APP)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	50
AuthenticationMethod (AM)	0 (No Encryption)
BulkBinaryThreshold (BBT)	32
BulkCharacterThreshold (BCT)	-1
BulkLoadBatchSize (BLBS)	1024
Charset (CS)	None
ConnectionReset (CR)	0 (Disabled)
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
Database (DB)	None
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
Description (n/a)	None
DistributedTransactionModel (DTM)	0 (XA Protocol)
EnableBulkLoad (EBL)	0 (Disabled)
EnableDescribeParam (EDP)	0 (Disabled)
EnableQuotedIdentifiers (EQI)	0 (Disabled)
EncryptionMethod (EM)	0 (No Encryption)
FailoverGranularity (FG)	0 (Non-Atomic)
FailoverMode (FM)	0 (Connection)
FailoverNetworkAddress (FNA)	None
FailoverPreconnect (FP)	0 (Disabled)
GSSClientLibrary (GS)	native
HostNameInCertificate (HNIC)	None

Table 12-2. Sybase Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
InitializationString (IS)	None
InterfacesFile (IF)	None
InterfacesFileServerName (IFSN)	None
KeyPassword (KP)	None
Keystore (KS)	None
KeystorePassword (KSP)	None
Language (LANG)	None
LoadBalanceTimeout (LBT)	0
LoadBalancing (LB)	0 (Disabled)
LoginTimeout (LT)	15
LogonID (UID)	None
MaxPoolSize (MXPS)	100
MinPoolSize (MNPS)	0
Network Address (NA)	None
OptimizePrepare (OP)	1 (Partial)
PacketSize (PS)	0
Password (PWD)	None
Pooling (POOL)	0 (Disabled)
QueryTimeout (QT)	0
RaiseErrorPositionBehavior (REPB)	0 (Default)
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
SelectMethod (SM)	0 (Cursor)
ServicePrincipalName (SPN)	None
TightlyCoupledDistributedTransactions (TCDT)	1 (Enabled)
TruncateTimeTypeFractions (TTTF)	0 (Disabled)
Truststore (TS)	None
TruststorePassword (TSP)	Password
ValidateServerCertificate (VSC)	1 (Enabled)

Table 12-2. Sybase Wire Protocol Attribute Names (cont.)

Attribute (Short Name)	Default
Workstation ID (WKID)	None
XAOpenStringParameters (XAOSP)	None

Alternate Servers

Attribute AlternateServers (ASVR)

Description A list of alternate database servers to which the driver will try to connect if the primary database server is unavailable. Specifying a value for this option enables connection failover for the driver. The value you specify must be in the form of a string that defines the physical location of each alternate server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection.

Valid Values `({NetworkAddress=`*`addressvalue`* `| InterfacesFileServerName=`*`sectionvalue`*`}[, ...])`

NetworkAddress and InterfacesFileServerName can be used in the same string.

NOTE: An alternate server address in IPv6 format must be enclosed in double quotation marks.

You must specify the network address of each alternate database server or the section in the Interfaces file that contains the network connection information for the Sybase database server you want to access (InterfacesFileServerName).

IMPORTANT: The alternate database servers specified for the driver must use the same network protocol as specified for the primary database server. On UNIX/Linux, only TCP/IP is

supported; on Windows, the driver supports Named Pipes and TCP/IP.

NOTE: The Alternate Servers option and the HA Failover Server Connection Information option are mutually exclusive.

Example The following example Alternate Servers values define four alternate database servers for connection failover:

```
(InterfacesFileServerName=Accounting, NetworkAddress=
\\MySybaseASE\Query, NetworkAddress="255.125.1.11, 4200",
NetworkAddress="SybaseASE2, 4200")
```

In this example, the network address of the last two alternates contain commas. In this case, enclose the network address with double quotation marks as shown.

Default None

GUI Tab Failover tab on page 623

Application Name

Attribute ApplicationName (APP)

Description The name used by Sybase to identify your application.

Valid Values *string*

where *string* is a valid application name.

Default None

GUI Tab Connection tab on page 619

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI Tab Advanced tab on page 615

Authentication Method

Attribute AuthenticationMethod (AM)

Description Specifies the method the driver uses to authenticate the user to the server when a connection is established. If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Valid Values 0 | 1 | 4

If set to 0 (No Encryption), the driver sends the user ID and password in clear text to the server for authentication.

If set to 1 (Encrypt Password), the driver sends the user ID in clear text and an encrypted password to the server for authentication.

If set to 4 (Kerberos), the driver uses Kerberos authentication. This method supports both Windows Active Directory Kerberos and MIT Kerberos environments.

If the specified authentication method is not supported by the database server, the connection fails and the driver generates an error.

Default 0 (No Encryption)

GUI Tab Security tab on page 617

Batch Size

Attribute BulkLoadBatchSize (BLBS)

Description The number of rows at a time that the driver sends to the database during bulk operations. This value applies to all methods of bulk loading.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | *x*

where *x* is the number of rows to send during a bulk operation.

Default 1024

GUI Tab Bulk tab on page 627

Bulk Binary Threshold

Attribute BulkBinaryThreshold (BBT)

Description The maximum size, in KB, of binary data that is exported to the bulk data file.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values -1 | 0 | *x*

where *x* is an integer that specifies the number of KB.

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 on page 627

Bulk Character Threshold

Attribute BulkCharacterThreshold (BCT)

Description 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.

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 on page 627

Charset

Attribute Charset (CS)

Description The name of a character set installed on the Sybase server to be used by the driver.

This option is not a substitute for the IANAAppCodePage option. See IANAAppCodePage for details.

Valid Values *charset*

where *charset* is the name of a character set installed on the Sybase server.

If unspecified, the character set setting on the Sybase server is used.

For the driver to return Unicode SQL types for connections to Sybase 12.5 and higher, use a value of UTF-8. Refer to the Sybase server documentation for a list of valid character sets.

Example If your client needs to receive data in iso-8859-1 from a non-Unicode Sybase server, you would specify a value of iso_1.

Default None

GUI Tab Connection tab on page 619

Connection Cache Size

Attribute CursorCacheSize (CCS)

Description The number of connections that the connection cache can hold.

Valid Values *x*

where *x* is a positive integer representing the number of connections that the connection cache can hold.

To enable the connection cache, you must set the Select Method option to 1 (enabled). Increasing the connection cache may increase performance of some applications but requires additional database resources.

Default 1

GUI Tab Performance tab on page 625

Connection Pooling

Attribute Pooling (POOL)

Description Specifies whether the driver uses connection pooling.

Determines whether the driver works with applications using multiple ODBC threads.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses connection pooling.

If set to 0 (Disabled), the driver does not use connection pooling.

Default 0 (Disabled)

GUI Tab Pooling tab on page 625

Connection Reset

Attribute ConnectionReset (CR)

Description Determines whether the state of connections that are removed from a pool for reuse by another application is reset to the initial configuration of the connection.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Pooling tab on page 625

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 623

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

If set to 0, there is no delay between retries.

If set to *x*, the driver waits between connection retry attempts the specified number of seconds.

Default 3

GUI Tab Failover tab on page 623

Cursor Positioning for Raiserror

Attribute RaiseErrorPositionBehavior (REPB)

Description Determines whether the driver returns raiserrors when the next statement is executed or handles them separately.

Valid Values 0 | 1

If set to 0 (Disabled), raiserrors are handled separately from surrounding statements. The error is returned when a raiserror is processed (for example, resulting from SQLExecute, SQLExecDirect, or SQLMoreResults). The result set is empty.

If set to 1 (Microsoft compatible), raiserrors are returned when the next statement is processed, and the cursor is positioned on the first row of the subsequent result set. This could result in multiple raiserrors being returned on a single execute.

Default 0 (Disabled)

GUI Tab Advanced tab on page 615

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 614

Database List

Attribute n/a

Description A list of database names that will appear in the drop-down list of the logon dialog box (see "Using a Logon Dialog Box" on page 637 for a description).

Valid Values *database_list*

where *database_list* is a comma-separated list of database names that will appear in the drop-down list of the logon dialog box.

Default None

GUI Tab Connection tab on page 619

Database Name

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 614

Description

Attribute: Description (n/a)

Description: 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 on page 614

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute: DefaultLongDataBuffLen (DLDBL)

Description: The maximum length of data (in KB) the driver can fetch from Long/LOB columns in a single round trip and the maximum length of data that the driver can send using the SQL_DATA_AT_EXEC parameter.

This option also applies to binding long parameters in chunks. The driver truncates any data passed in a Long/LOB SQL_DATA_AT_EXEC parameter to the size specified.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values: An integer in multiples of 1024

The value must be in multiples of 1024 (for example, 1024, 2048). You need to increase the default value if the total size of any Long data exceeds 1 MB. This value is multiplied by 1024 to determine the total maximum length of fetched data. For

example, if you enter a value of 2048, the maximum length of data would be 1024 x 2048, or 2097152 (2 MB).

Default 1024

GUI Tab Advanced tab on page 615

Distributed Transaction Model

Attribute DistributedTransactionModel (DTM)

Description The model to use for distributed transaction support. The driver supports two different models: XA Protocol and Native OLE.

Valid Values 0 | 1

Specify the appropriate distributed transaction protocol, either 0 (XA Protocol) or 1 (Native OLE)

Default 0 (XA Protocol)

GUI Tab Advanced tab on page 615

Enable Bulk Load

Attribute EnableBulkLoad (EBL)

Description Specifies the bulk load method.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses the database bulk load protocols. If the protocols cannot be used, the driver returns an error.

If set to 0 (Disabled), the driver uses standard parameter arrays.

Default 0 (Disabled)

GUI Tab Bulk tab on page 627

Enable Describe Parameter

Attribute EnableDescribeParam (EDP)

Description 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

If set to 1 (Enabled), the driver supports SQLDescribeParam. The Prepare Method option must be set to 0 or 1, and the SQL statement must not include long parameters. If using Microsoft Remote Data Objects (RDO) to access data, you must use this value.

If set to 0 (Disabled), the driver does not support SQLDescribeParam.

Default 0 (Disabled)

GUI Tab Advanced tab on page 615

Enable Quoted Identifiers

Attribute EnableQuotedIdentifiers (EQI)

Description Determines whether the driver supports the use of quoted identifiers.

Valid Values 0 | 1

If set to 1 (Enabled), the driver supports the use of quoted identifiers. Double quotation marks (") must be used to enclose identifiers, such as column and table names. Character strings must be enclosed in single quotation marks, for example:

```
SELECT "au_id"
FROM "authors"
WHERE "au_lname" = 'O''Brien'
```

If set to 0 (Disabled), the driver does not support the use of quoted identifiers and generates an error when quoted identifiers are encountered.

Default 0 (Disabled)

GUI Tab Advanced tab on page 615

Encryption Method

Attribute EncryptionMethod (EM)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

If set to 0 (No Encryption), data is not encrypted.

If set to 1 (SSL), data is encrypted using SSL. If the server supports protocol negotiation, the driver and server negotiate the use of TLS1, SSL3, or SSL2 in that order.

This option can only be set to 1 when Authentication Method is set to 0 or 1.

Default 0 (No Encryption)

GUI Tab Security tab on page 617

Failover Granularity

Attribute FailoverGranularity (FG)

Description 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

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 on page 623

Failover Mode

Attribute FailoverMode (FM)

Description The type of failover method the driver will use.

The Alternate Servers option specifies one or multiple alternate servers for failover and is required for all failover methods.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1 | 2

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 on page 623

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

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 on page 623

Fetch Array Size

Attribute ArraySize (AS)

Description The number of rows the driver retrieves from the server for a fetch. This is not the number of rows given to the user. This connection option can affect performance.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values *x*

where *x* is a positive integer specifying the number of bytes.

Default 50

GUI Tab Performance tab on page 621

GSS Client Library

Attribute GSSClientLibrary (GS)

Description The name of the GSS client library that the driver uses to communicate with the Key Distribution Center (KDC).

The driver uses the standard path for loading the specified client library.

Valid Values native | *client_library*

where *client_library* is a GSS client library installed on the client.

If set to *client_library*, the driver uses the specified GSS client library.

If set to native, the driver uses the GSS client shipped with the operating system.

Default native

GUI Tab Security tab on page 617

HA Failover Server Connection Information/Network Address

Attribute FailoverNetworkAddress (FNA)

Description The network address of the High Availability (HA) Failover server to be used in the event of a connection loss. The driver detects the dropped connection and automatically reconnects to the specified HA Failover server. This option is valid only for Sybase 12 and higher servers that have the High Availability Failover feature enabled.

Valid Values *IP_address*, *port_number* | *pipe_address*, *port_number* | *server_name*, *port_number*

where

IP_ address is the IP address that uniquely identifies the HA Failover server.

port_number is the port number assigned to the listener process on the HA Failover server.

server_name is a name that uniquely identifies the HA Failover server. You can use this format if your environment supports named servers.

pipe_address is the pipe address of the HA Failover server. This format is required if using NamedPipes as the network protocol.

NOTE: The HA Failover Server Connection Information option and the Alternate Servers option are mutually exclusive.

Example

```
199.226.224.34, 5000
```

or

```
\\machine1\sybase\pipe\query, 5000
```

or

```
Sybaseserver, 5000
```

Default None

GUI Tab Failover tab on page 623

Host Name In Certificate

Attribute HostNameInCertificate (HNIC)

Description A host name for certificate validation when SSL encryption is enabled (Encryption Method=SSL) and validation is enabled (Validate Server Certificate=1). This option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

Valid Values *host_name* | #SERVERNAME#

where the *host_name* is the host name specified in the certificate. Consult your SSL administrator for the correct value.

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 Common Name parts.

If set to #SERVERNAME#, then the driver compares the host server name specified as part of a data source or connection string to the dnsName value or the commonName.

Default None

GUI Tab Security tab on page 617

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 “Internationalization, Localization,

and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 615

Initialization String

Attribute InitializationString (IS)

Description A SQL command that is issued immediately after connecting to the database to manage session settings.

NOTE: If the statement fails to execute, the connection fails and the driver reports the error returned from the server.

Valid Values *SQL_command*

where *SQL_command* is a valid SQL command that is supported by the database.

Example To allow delimited identifiers, specify:

```
Initialization String=set QUOTED_IDENTIFIER on
```

Default None

GUI Tab Advanced tab on page 615

Interfaces File

Attribute InterfacesFile (IF)

Description The directory to the Interfaces file.

NOTE: This option and the Network Address option are mutually exclusive.

Valid Values *file_dir*

where *file_dir* is the directory to the Interfaces file.

If unspecified and a value is specified for the Server Name option, the driver looks for the path name of the Interfaces file in the Registry under HKEY_LOCAL_MACHINE\SOFTWARE\DataDirect\InterfacesFile. If this Registry value is empty, the driver will try to open the SQL.INI file found in the same directory where the driver is located and use it as the Interfaces file.

Default None

GUI Tab General tab on page 614

Key Password

Attribute KeyPassword (KP)

Description The password used to access the individual keys in the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. Keys stored in a keystore can be individually password-protected. To extract the key from the keystore, the driver must have the password of the key.

Valid Values *key_password*

where *key_password* is the password of a key in the keystore.

Default None

GUI Tab Security tab on page 617

Keystore

Attribute Keystore (KS)

Description The directory of the keystore file to be used when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file.

Valid Values *keystore_directory*

where *keystore_directory* is the location of the keystore file.

Default None

GUI Tab Security tab on page 617

Keystore Password

Attribute KeystorePassword (KSP)

Description The password used to access the keystore file when SSL is enabled (Encryption Method=1) and SSL client authentication is enabled on the database server. The keystore file contains the certificates that the client sends to the server in response to the server's certificate request.

NOTE: The keystore and truststore files may be the same file; therefore, they may have the same password.

Valid Values *keystore_password*

where *keystore_password* is the password of the keystore file.

Default None

GUI Tab Security tab on page 617

Language

Attribute Language (LANG)

Description The national character set installed on the Sybase server.

Valid Values *charset*

where *charset* is the national character set installed on the Sybase server.

Default None (English)

GUI Tab Connection tab on page 619

Load Balance Timeout

Attribute: LoadBalanceTimeout (LBT)

Description: The number of seconds to keep inactive connections open in a connection pool.

NOTE: The Min Pool Size option may cause some connections to ignore this value.

Determines whether the driver works with applications using multiple ODBC threads.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values: 0 | *x*

where *x* is a positive integer.

If set to 0, inactive connections are kept open.

If set to *x*, inactive connections are closed after the specified number of seconds passes.

Default: 0 (Disabled)

GUI Tab: Pooling tab on page 625

Load Balancing

Attribute: LoadBalancing (LB)

Description: Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values: 0 | 1

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).

NOTE: This option has no effect unless alternate servers are defined for the Alternate Servers connection option.

Default 0 (Disabled)

GUI Tab Failover tab on page 623

Login Timeout

Attribute LoginTimeout (LT)

Description 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 statement, set a different value in the SQL_ATTR_LOGIN_TIMEOUT statement attribute.

Valid Values -1 | 0 | *x*

where *x* is a positive integer that specifies a number of seconds.

If set to -1, the connection request does not time out. The driver silently ignores the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

If set to 0, the connection request does not time out, but the driver responds to the SQL_ATTR_LOGIN_TIMEOUT attribute on the SQLSetConnectAttr() function.

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 on the SQLSetConnectAttr() function.

Default 15

GUI Tab Advanced tab on page 615

Max Pool Size

Attribute MaxPoolSize (MXPS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values An integer from 1 to 65535

For example, if set to 20, the maximum number of connections allowed in the pool is 20.

Default 100

GUI Tab Pooling tab on page 625

Min Pool Size

Attribute MinPoolSize (MNPS)

Description The minimum number of connections that are opened and placed in a connection pool, in addition to the active connection, when the pool is created. The connection pool retains this

number of connections, even when some connections exceed their Load Balance Timeout value.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | *x*

where *x* is an integer from 1 to 65535.

For example, if set to 5, the start-up number of connections in the pool is 5 in addition to the current existing connection.

If set to 0, no connections are opened in addition to the current existing connection.

Default 0

GUI Tab Pooling tab on page 625

Network Address

Attribute Network Address (NA)

Description A unique identifier assigned to the Sybase server machine.

NOTE: This option is mutually exclusive with the Interfaces File and the Server Name option.

Valid Values *server_name* | *IP_address*

where

server_name is the Sybase server name specified as: *named_server*, *port_number*. For example, you can enter `SSserver, 5000`.

IP_address is the Sybase server address specified as: *IP_address*, *port_number*. For example, you can enter `199.226.224.34, 5000`. The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details about these formats.

Default

None

GUI Tab General tab on page 614

Packet Size

Attribute PacketSize (PS)

Description Determines the number of bytes for each database protocol packet that is transferred from the database server to the client machine. Adjusting the packet size can improve performance. The optimal value depends on the typical size of data that is inserted, updated, or returned by the application and the environment in which it is running. Typically, larger packet sizes work better for large amounts of data. For example, if an application regularly returns character values that are 10,000 characters in length, using a value of 32 (16 KB) typically results in improved performance.

NOTE: The ODBC connection option SQL_PACKET_SIZE provides the same functionality as the Packet Size option; however SQL_PACKET SIZE and the Packet Size option are mutually exclusive. If Packet Size is specified, the driver returns the message `Driver Not Capable` if an application attempts to call SQL_PACKET_SIZE. If you do not set the Packet Size option, application calls to SQL_PACKET_SIZE are accepted by the driver.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values -1 | 0 | *x*

If set to -1, the driver uses the maximum packet size that is set by the database server.

If set to 0, the driver uses the default packet size that is used by the database server.

If set to *x*, an integer from 1 to 127, the driver uses a packet size that is a multiple of 512 bytes. For example, PacketSize=8 means to set the packet size to 8 * 512 bytes (4096 bytes).

NOTE: IF SSL encryption is used, the driver must use the packet size that is specified by the server. Any value set for this option or the SQL_PACKET_SIZE connect option is ignored if SSL encryption is used.

Default 0

GUI Tab Performance tab on page 625

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Prepare Method

Attribute OptimizePrepare (OP)

Description Determines whether stored procedures are created on the server for calls to SQLPrepare.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1 | 2 | 3

If set to 0 - None, stored procedures are created for every call to SQLPrepare. This setting can result in decreased performance when processing statements that do not contain parameters.

If set to 1 - Partial, the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and run directly at the time of SQLExecute.

If set to 2 - Full, stored procedures are never created. The driver caches the statement, executes it directly at the time of SQLExecute, and reports any syntax or similar errors at the time of SQLExecute.

If set to 3 - Full at Prepare, stored procedures are never created. This is identical to value 2 except that any syntax or similar errors are returned at the time of SQLPrepare instead of SQLExecute. Use this setting only if you must have syntax errors reported at the time of SQLPrepare.

Default 1 (Partial)

GUI Tab Performance tab on page 625

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.

Valid Values -1 | 0 | *x*

where *x* is a positive integer representing the number of seconds.

If set to -1, the query does not time out. The driver silently ignores the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

If set to 0, the query does not time out, but the driver responds to the SQL_ATTR_QUERY_TIMEOUT attribute on the SQLSetStmtAttr() function.

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 on the SQLSetStmtAttr() function.

Default 0

GUI Tab Advanced tab on page 615

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 615

Select Method

Attribute SelectMethod (SM)

Description Determines whether database cursors are used for Select statements.

This connection option can affect performance. See "Performance Considerations" on page 680 for details.

Valid Values 0 | 1

If set to 0 (Cursor), database cursors are used. In some cases performance degradation can occur when performing large numbers of sequential Select statements because of the amount of overhead associated with creating database cursors.

If set to 1 (Direct), Select statements are run directly without using database cursors, and the data source is limited to one active statement.

Default 0 (Cursor)

GUI Tab Performance tab on page 625

Server Name

Attribute InterfacesFileServerName (IFSN)

Description The name of the section in the Interfaces file containing the network connection information for the Sybase server. Typically, the section name is the host name of the Sybase server.

NOTE: The Network Address option and the Server Name option are mutually exclusive.

Valid Values *section_name*

where *section_name* is a section in the Interfaces file containing the network connection information for the Sybase server.

Default None

GUI Tab General tab on page 614

Service Principal Name

Attribute ServicePrincipalName (SPN)

Description The service principal name to be used by driver for Kerberos authentication.

Valid Values *servicePrincipalName*

where *servicePrincipalName* is a valid service principal name.

If unspecified, the value of the Network Address option is used as the service principal name. If Authentication Method is set to 0 or 1, the value of the Service Principal Name option is ignored.

Default None

GUI Tab Security tab on page 617

Tightly Coupled Distributed Transactions

Attribute TightlyCoupledDistributedTransactions (TCDT)

Description Sybase 12 or higher server only. Determines whether the driver ensures that multiple connections within the same distributed transaction obey other's locks.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses tightly coupled distributed transactions. Multiple connections within the same distributed transaction obey other's locks.

If set to 0 (Disabled), the driver does not use tightly coupled distributed transactions. Multiple connections within the same distributed transaction may hang each other because the connections do not obey other's locks. This value can provide better performance if concurrency of data is not needed.

Default 1 (Enabled)

GUI Tab Advanced tab on page 615

Truncate Time Type Fractions

Attribute TruncateTimeTypeFractions (TTTF)

Description Sybase 12.5.1 and higher only. Determines whether the driver sets fractional seconds to zero (0) when converting data from the TIME data type to TIMESTAMP, CHAR, or WCHAR data types.

Valid Values 0 | 1

If set to 1 (Enabled), the driver converts fractional seconds to zero when converting the TIME data type.

If set to 0 (Disabled), the driver does not set fractional seconds to zero when converting the TIME data type.

Default 0 (Disabled)

GUI Tab Advanced tab on page 615

Truststore

Attribute Truststore (TS)

Description The directory of the location of the truststore file to be used when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

NOTE: The truststore and keystore files may be the same file.

Valid Values *truststore_directory*

where *truststore_directory* is the directory where the truststore file is located.

Default None

GUI Tab Security tab on page 617

Truststore Password

Attribute TruststorePassword (TSP)

Description The password that is used to access the truststore file when SSL is enabled (Encryption Method=1) and server authentication is used. The truststore file contains a list of the Certificate Authorities (CAs) that the client trusts.

NOTE: The truststore and keystore files may be the same file; therefore, they may have the same password.

Valid Values *truststore_password*

where *truststore_password* is a valid password for the truststore file.

Default None

GUI Tab Security tab on page 617

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 on page 617

Validate Server Certificate

Attribute ValidateServerCertificate (VSC)

Description Determines whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled (Encryption Method=1). When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority (CA). Allowing the driver to trust any certificate returned from the server even if the issuer is not a trusted CA is useful in test environments because it eliminates the need to specify truststore information on each client in the test environment.

Truststore information is specified using the Trust Store and Trust Store Password options.

Valid Values 0 | 1

If set to 1 (Enabled), the driver validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted CA in the truststore file. If the Host Name In Certificate option is specified, the driver also validates the certificate using a host name. The Host Name In Certificate option provides additional security against man-in-the-middle (MITM) attacks by ensuring that the server the driver is connecting to is the server that was requested.

If set to 0 (Disabled), the driver does not validate the certificate that is sent by the database server. The driver ignores any truststore information specified by the Trust Store and Trust Store Password options.

Default 1 (Enabled)

GUI Tab Security tab on page 617

Workstation ID

Attribute Workstation ID (WKID)

Description An identifier for the client machine.

Valid Values *ID*

where *ID* is workstation ID use by the client machine.

Default None

GUI Tab Connection tab on page 619

XA Open String Parameters

Attribute XAOpenStringParameters (XAOSP)

Description Determines the name of trace files generated for XA open string parameters.

Valid Values `-Ltrace_filename`

where `trace_filename` is a string that identifies trace files generated for XA open string parameters. If specified, two trace files are created. The first trace file traces all XA call activities and is named exactly as specified. The second trace file traces any enlistment and unenlistment procedures and is named as specified with a "driver" extension.

Example If you specify `-LXAtrace`, the driver creates two trace files: XAtrace and XAtrace.driver.

Default None

GUI Tab Advanced tab on page 615

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Connection Pooling (ConnectionPooling): If you enable the driver to use connection pooling, you can set additional options that affect performance:

- **Load Balance Timeout**: 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**: 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**: 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**: A connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size, if one has been specified. The connection pool retains this number of connections, even when some connections exceed their Load Balance Timeout value.

Default Buffer Size for Long/LOB Columns (DefaultLongDataBuffLen): To improve performance when your application fetches images, pictures, or long text or binary data, a buffer size can be set to accommodate the maximum size of the data. The buffer size should only be large enough to accommodate the maximum amount of data retrieved; otherwise, performance is reduced by transferring large amounts of data into an oversized buffer. If your application retrieves more than 1 MB of data, the buffer size should be increased accordingly.

Enable Bulk Load (EnableBulkLoad): If your application performs bulk loading of data, you can improve performance by configuring the driver to use the database system's bulk load functionality instead of database array binding. The trade-off to

consider for improved performance is that using the bulk load functionality can bypass data integrity constraints.

Encryption Method (EncryptionMethod): Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) that is required to encrypt and decrypt data.

Failover Mode (FailoverMode): Although high availability that replays queries after a failure provides increased levels of protection, it can adversely affect performance because of increased overhead.

Fetch Array Size (ArraySize): If the Select Method connection option is set to 0 and your application fetches more than 50 rows at a time, you should set Fetch Array Size to the approximate number of rows being fetched. This reduces the number of round trips on the network, thereby increasing performance. For example, if your application normally fetches 200 rows, it is more efficient for the driver to fetch 200 rows at one time over the network than to fetch 50 rows at a time during four round trips over the network. You should use Fetch Array Size in conjunction with Select Method.

NOTE: The ideal setting for your application will vary. To calculate the ideal setting for this option, you must know the size in bytes of the rows that you are fetching and the size in bytes of your Network Packet. Then, you must calculate the number of rows that will fit in your Network Packet, leaving space for packet overhead. For example, suppose your Network Packet size is 1024 bytes and the row size is 8 bytes. Dividing 1024 by 8 equals 128; however, the ideal setting for Fetch Array Size is 127, not 128, because the number of rows times the row size must be slightly smaller than the Network Packet size.

Packet Size (PacketSize): Typically, it is optimal for the client to use the maximum packet size that the database server allows. This reduces the total number of round trips required to return data to the client, thus improving performance. Therefore,

performance can be improved if the PacketSize attribute is set to the maximum packet size of the Sybase ASE server.

Prepare Method (OptimizePrepare): If your application executes the same SQL statements multiple times, performance can be improved by creating a stored procedure on the server at prepare time. If your application executes one of these prepared statements multiple times, performance will improve because the driver created a stored procedure and executing a stored procedure is faster than executing a single SQL statement; however, if a prepared statement is only executed once or is never executed, performance can decrease. If your application executes the same SQL statements multiple times, the Prepare Method option should be set to 1.

Select Method (SelectMethod): If your application often executes a SQL statement before processing or closing the previous result set, then it uses multiple active statements per connection. The default setting (0) of this option causes the driver to use database cursors for Select statements and allows an application to process multiple active statements per connection. An active statement is defined as a statement where all the result rows or result sets have not been fetched. This can cause high overhead on the server. If your application does not use multiple active statements, however, setting Select Method to 1 will increase performance of Select statements by allowing the server to return results without using a database cursor. If this option is set to 0, it should be used in conjunction with Fetch Array Size (ArraySize). If this option is set to 1, Fetch Array Size (ArraySize) has no effect.

Data Types

Table 12-3 shows how the Sybase data types are mapped to the standard ODBC data types. "Unicode Support" on page 685 lists Sybase to Unicode data type mappings.

Table 12-3. Sybase Data Types

Sybase	ODBC
BIGINT[1]	SQL_BIGINT
BINARY	SQL_BINARY
BIT	SQL_BIT
CHAR	SQL_CHAR
DATE[2]	SQL_TYPE_DATE
DATETIME	SQL_TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
FLOAT	SQL_FLOAT
IMAGE	SQL_LONGVARBINARY
INT	SQL_INTEGER
MONEY	SQL_DECIMAL
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLDATETIME	SQL_TYPE_TIMESTAMP
SMALLINT	SQL_SMALLINT
SMALLMONEY	SQL_DECIMAL
SYSNAME	SQL_VARCHAR
TEXT	SQL_LONGVARCHAR
TIME[2]	SQL_TYPE_TIME
TIMESTAMP	SQL_VARBINARY
TINYINT	SQL_TINYINT
UNSIGNED BIGINT[1]	SQL_BIGINT
UNSIGNED INT[1]	SQL_INTEGER

Table 12-3. Sybase Data Types (cont.)

UNSIGNED SMALLINT [1]	SQL_SMALLINT
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_VARCHAR

1. Sybase 15 only.
2. Sybase 12.5.1 and higher only.

NOTE FOR USERS OF SYBASE 12.5 AND HIGHER: The Sybase Wire Protocol driver supports extended new limits (XNL) for character and binary columns—columns with lengths greater than 255.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Unicode Support

The Sybase Wire Protocol driver maps the Sybase data types to Unicode data types as shown in the following table:

Sybase Data Type	Mapped to. . .
CHAR [1]	SQL_WCHAR
SYSNAME [1]	SQL_VARCHAR
TEXT [1]	SQL_WLONGVARCHAR
UNICHAR [2]	SQL_WCHAR
UNITEXT [3]	SQL_WLONGVARCHAR
UNIVARCHAR [2]	SQL_WVARCHAR
VARCHAR [1]	SQL_WVARCHAR

1. This data type is available only if the data source is configured to use the UTF-8 character set.

2. On Sybase 12.5 servers, this data type is available only if the data source is configured to use the UTF-8 character set. On Sybase 12.5.1 and higher servers, this data type is always available, even if the data source is not configured to use the UTF-8 character set.
3. This data type is available on Sybase 15 and higher servers only.

For data types that require the UTF-8 character set, see Table 12-2 on page 639 for more information about the Charset connection string attribute.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Unexpected Characters

Users are sometimes surprised when they insert a character into a database, only to have a different character displayed when they fetch it from the database. There are many reasons this can happen, but it most often involves code page issues, not driver errors.

Client and server machines in a database system each use code pages, which can be identified by a name or a number, such as Shift_JIS (Japanese) or cp1252 (Windows English). A code page is a mapping that associates a sequence of bits, called a code point, with a specific character. Code pages include the characters and symbols of one or more languages. Regardless of geographical

location, a machine can be configured to use a specific code page. Most of the time, a client and database server would use similar, if not identical, code pages. For example, a client and server might use two different Japanese code pages, such as Shift_JIS and EUC_JP, but they would still share many Japanese characters in common. These characters might, however, be represented by different code points in each code page. This introduces the need to convert between code pages to maintain data integrity. In some cases, no one-to-one character correspondence exists between the two code points. This causes a substitution character to be used, which can result in displaying an unexpected character on a fetch.

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage attribute (see "IANAAppCodePage" on page 661). If it does not find a specific setting for IACP, it defaults to a value of ISO_8859_1.

If the client and server code pages are compatible, the driver transmits data in the code page of the server. Even though the pages are compatible, a one-to-one correspondence for every character may not exist. If the client and server code pages are completely dissimilar, for example, Russian and Japanese, then many substitutions occur because very few, if any, of the characters are mapped between the two code pages.

The following is a specific example of an unexpected character:

- The Windows client machine is running code page cp1252.
- The Sybase server is running code page cp850.
- You insert decimal literals for character data. You think you are inserting LATIN SMALL LETTER I WITH ACUTE (í) and BOX DRAWINGS DOUBLE VERTICAL (‖) in the database. When you fetch the data, you see INVERTED EXCLAMATION MARK (¡) and MASCULINE ORDINAL INDICATOR (º) displayed on the client instead.

This occurs because the code points do not correspond in the two code pages. An example of syntax you would use to insert the decimal literals is:

```
CREATE table cp850chars(val text )
INSERT INTO cp850chars values( CHAR(161)+CHAR(186))
```

This effectively inserts the hexadecimal bytes for the numbers 161 (0xA1) and 186 (0xBA) into the text column. Each of these hexadecimal bytes is treated as the single byte code point for the character it represents. The problem is that the character representation for these two particular hexadecimal values is different from code page cp850 to code page cp1252. On cp850, these hexadecimal values represent í (0xA1) and ‖ (0xBA), which is what you thought you were inserting by using the previously described syntax. When you fetch these hexadecimal values, however, the characters displayed on your client machine are ¡ (0xA1) and º (0xBA), because that is what the hexadecimal values represent in code page cp1252. This is not a matter of data corruption or substitution; these hexadecimal values simply represent different values in the two different code pages.

This is not a driver error. It occurs because the code points map differently and because some characters do not exist in a code page. The best way to avoid these problems is to use the same code page on both the client and server machines.

MTS Support

On Windows, the driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC) using the XA Protocol. For a general discussion of MTS and DTC, refer to the help file of the Microsoft Transaction Server SDK.

NOTE: The DataDirect Connect *for* ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 *for* ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

To enable DTC support, you must be accessing Sybase Adaptive Server Enterprise 12.0 or higher. You can choose either Native OLE and XA protocol distributed transactions. See the Distributed Transaction Model option documented in "Configuring and Connecting to Data Sources" on page 609 for details.

To enable distributed transaction in the Sybase server:

1. Assign the dtm_tm_role to each user who will participate in distributed transactions (who will log in to Adaptive Server). You can do this using the sp_role command. For example:

   ```
   sp_role "grant", dtm_tm_role, user_name
   ```

   In the open string for resource managers, the specified username must have the dtm_tm_role.

2. Specify a default database other than the master for each user. Sybase cannot start distributed transactions in a master database.

NULL Values

When the Sybase Wire Protocol driver establishes a connection, the driver sets the Sybase database option ansinull to on. Setting ansinull to on ensures that the driver is compliant with the ANSI SQL standard, which makes developing cross-database applications easier.

By default, Sybase does not evaluate NULL values in SQL equality (=), inequity (<>), or aggregate function comparisons in an ANSI SQL-compliant manner. For example, the ANSI SQL specification defines that `col1=NULL` always evaluates to false:

```
SELECT * FROM table WHERE col1 = NULL
```

Using the default database setting (ansinull=off), the same comparison evaluates to true instead of false.

Setting ansinull to on changes the default database behavior so that SQL statements must use `IS NULL` instead of `=NULL`. For example, using the Sybase Wire Protocol driver, if the value of col1 in the following statement is NULL, the comparison evaluates to true:

```
SELECT * FROM table WHERE col1 IS NULL
```

In your application, you can restore the default Sybase behavior for a connection in the following ways:

- Use the Initialization String option to specify the SQL command `set ANSINULL off`. For example, the following connection string ensures that the handling of NULL values is restored to the Sybase default for the current connection:

  ```
  DSN=SYB TABLES;DB=PAYROLL;IS=set ANSINULL off
  ```

- Explicitly execute the following statement after the connection is established:

  ```
  SET ANSINULL OFF
  ```

Using DataDirect Bulk Load on Sybase

DataDirect Bulk Load offers a simple, consistent way to do bulk load operations. See "Using DataDirect Bulk Load" on page 115 for details.

For Sybase, some additional database configuration is required when the destination table for a bulk load operation does not have an index defined. If you are using a destination table that does not have an index defined, you can ask the database operator to execute the following commands:

```
use master
sp_dboption test, "select into/bulkcopy/pllsort", true
```

This option is required to perform operations that do not keep a complete record of the transaction in the log. For more information, refer to the Sybase documentation.

Alternatively, you can define an index on the destination table.

Failure to properly configure the database results in errors such as the following:

```
"You cannot run the non-logged version of bulk copy in this
database. Please check with the DBO."
```

Bulk Copy Operations and Transactions

Sybase does not support a bulk insert within a transaction, and returns an error if a bulk copy operation is attempted in the scope of an existing transaction.

The Sybase server treats each batch of the bulk copy operation as a single transaction. If any rows in the batch are rejected, the entire transaction is rolled back.

Performance Considerations

Sybase defines two bulk copy modes, described in Table 12-4. Sybase automatically selects the appropriate mode at run time. For more information, refer to your Sybase documentation.

Table 12-4. Summary of Fast and Slow Bulk Copy Mode Characteristics

Characteristic	Fast Bulk Copy Mode	Slow Bulk Copy Mode
Destination Table Characteristics	No indexes or triggers on destination table	One or more indexes or triggers
Database Configuration Required	The into/bulkcopy/pllsort dboption must be set to true.	None
Logging Performed	Page allocations are logged, but row inserts are not	Row inserts are logged
Transaction Log Handling	You must dump the database before backing up (dumping) the transaction log.	The transaction log can become very large. After the bulk copy completes, back up your database with dump database, then truncate the log with dump transaction.

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

The Sybase database system supports isolation levels 0 (read uncommitted), 1 (read committed, the default), 2 (repeatable read), and 3 (serializable). It supports page-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The following functions are supported:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Sybase Wire Protocol driver.

Number of Connections and Statements Supported

The Sybase database system supports multiple connections and multiple statements per connection. If the Select Method option on the Performance tab or the connection string attribute SelectMethod is set to 1 (Direct), Sybase data sources are limited to one active statement in manual commit mode.

Using Arrays of Parameters

When designing an application, using parameter arrays for bulk inserts or updates, for example, can improve performance. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

Because Sybase databases do not support parameter arrays natively, the Sybase Wire Protocol driver emulates them by sending T-SQL batches of Insert or Update statements to the database, which will improve performance.

13 The Oracle Driver

The DataDirect Connect Series *for* ODBC Oracle driver (the Oracle driver) is available in both 32- and 64-bit versions and supports the following Oracle database servers when using the appropriate client software:

- Oracle 11*g* R1 (11.1)
- Oracle 10*g* R1 and R2 (10.1 and 10.2)
- Oracle 9*i* R1 and R2 (9.0.1 and 9.2)
- Oracle 8*i* R3 (8.1.7)
- Oracle 8*i* R1 and R2 (8.1.5 and 8.1.6) [32-bit only]
- Oracle 8.0.5 and higher [32-bit only]

The Oracle driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Oracle driver.

NOTE: The Oracle driver requires Oracle client software. DataDirect Technologies also provides an Oracle driver that does not require any client software to access Oracle databases. See "The Oracle Wire Protocol Driver" on page 385 for details.

Driver Requirements

This section provides the system requirements for using the Oracle driver on Windows, UNIX, and Linux.

IMPORTANT: You must have all components of the Oracle client software installed; otherwise, the driver will not operate properly. You must have the appropriate DLLs or shared libraries and objects on your path.

Although an earlier version of a client can access a later version of a database, for example, client 9*i* to server 10*g*, to ensure that you have access to all of the features of a particular database, you should use the client that matches the database version, for example, client 10*g* to server 10*g*.

NOTE: The Oracle driver supports Oracle 10*g* clients; however, the clients are not available for all operating systems supported by the driver. Consult the Oracle Web site for current client availability.

Windows

For 32-bit drivers, Oracle Net8 Client 8.1.6 or higher is required.

For 64-bit drivers, Oracle client software 9.2.0.2.1 or higher is required on Itanium II. Oracle client software 10.1 or higher is required on x64.

UNIX and Linux

For 32-bit drivers, Oracle Net8 Client 8.1.6 or higher is required (8.1.6.1 on Linux).

For 64-bit drivers, Oracle client software 9*i* R2 or higher is required on Linux for Itanium II and UNIX. Oracle client software 10.1 or higher is required for Linux on x64.

Before you can use the Oracle driver, you must have a supported Oracle client installed on your workstation in the $ORACLE_HOME source tree. ORACLE_HOME is an environment

variable created by the Oracle installation process that identifies the location of your Oracle client components.

Set the environment variable ORACLE_HOME to the directory where you installed the Oracle client. For example, for C-shell users, the following syntax is valid:

```
setenv ORACLE_HOME /databases/oracle
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
ORACLE_HOME=/databases/oracle;export ORACLE_HOME
```

32-bit drivers—Building the Required Oracle Net8 Shared Library on HP-UX 11

You must build a replacement shared library for Oracle Net8 Client 8.1.6 on HP-UX 11. This shared library, libclntsh.sl, contains your unique Oracle Net8 configuration, which is used by the Oracle driver to access local and remote Oracle databases.

The shared library libclntsh.sl is built by the Oracle script genclntsh. The genclntsh script provided by Oracle causes errors resulting from undefined symbols. Run the genclntsh816 script provided by DataDirect Technologies to build a replacement libclntsh.sl. This script, in the src/oracle directory, places the new libclntsh.sl in ../../lib, which is your $ODBC_HOME/lib directory; it does not overwrite the original libclntsh.sl in the $ORACLE_HOME/lib directory.

Before you build the Oracle Net8 shared library, install Oracle and set the environment variable ORACLE_HOME to the directory where you installed Oracle.

For Oracle Net8 Client 8.1.6 on HP-UX 11, the following commands build the Oracle Net8 shared library:

```
cd ${ODBC_HOME}/src/oracle
genclntsh816
```

WARNING: The $ODBC_HOME/lib directory, containing the correct libclntsh library, *must* be on the SHLIB_PATH *before* $ORACLE_HOME/lib. Otherwise, the original Oracle library will be loaded, resulting in the unresolved symbol error.

Connecting to Oracle 8.1.7 from HP-UX

To connect to Oracle 8.1.7 from HP-UX, you must have the HP patch PHSS_22514 installed on the operating system, and you must set the LD_PRELOAD system variable to the absolute path of the libjava.sl library.

Advanced Features

The driver supports the following advanced features:

- Failover
- Client Load Balancing
- Connection Retry

See "Advanced Features" on page 85 for a general description of these features and their configuration requirements. See the specific tabs associated with these features in the driver Setup dialog box for information about individual connection options.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 “Quick Start Connect” on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See “Using a Connection String” on page 710 and Table 13-1 on page 713 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See “Environment Configuration” on page 45 for basic setup information and “Environment Variables” on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See “Configuration Through the Administrator” on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 13-1 on page 713 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

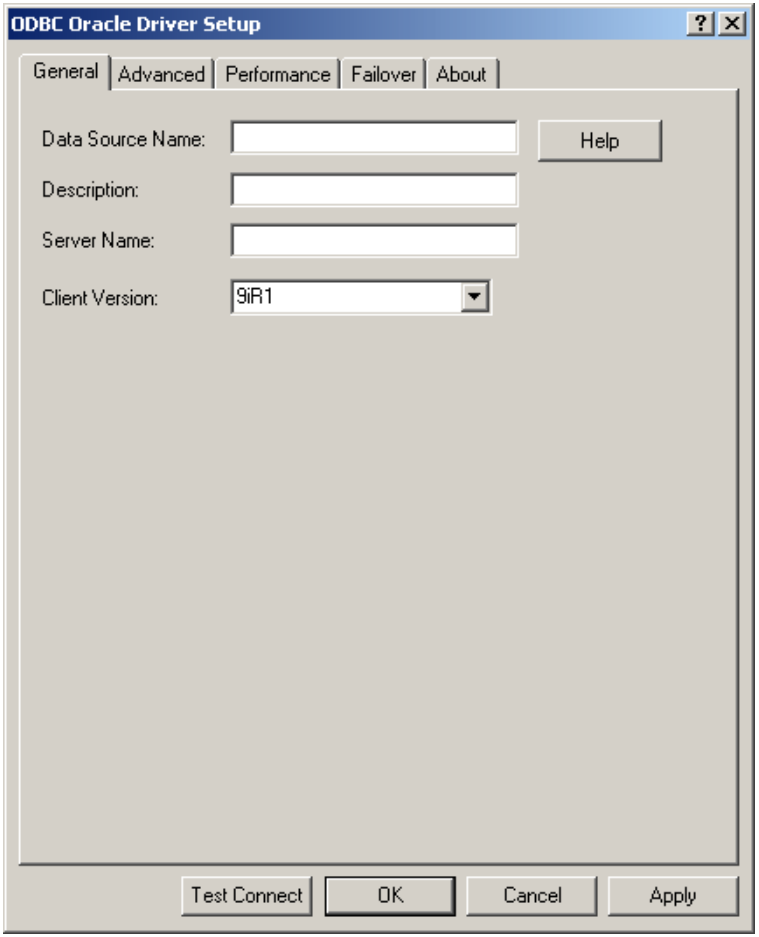
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

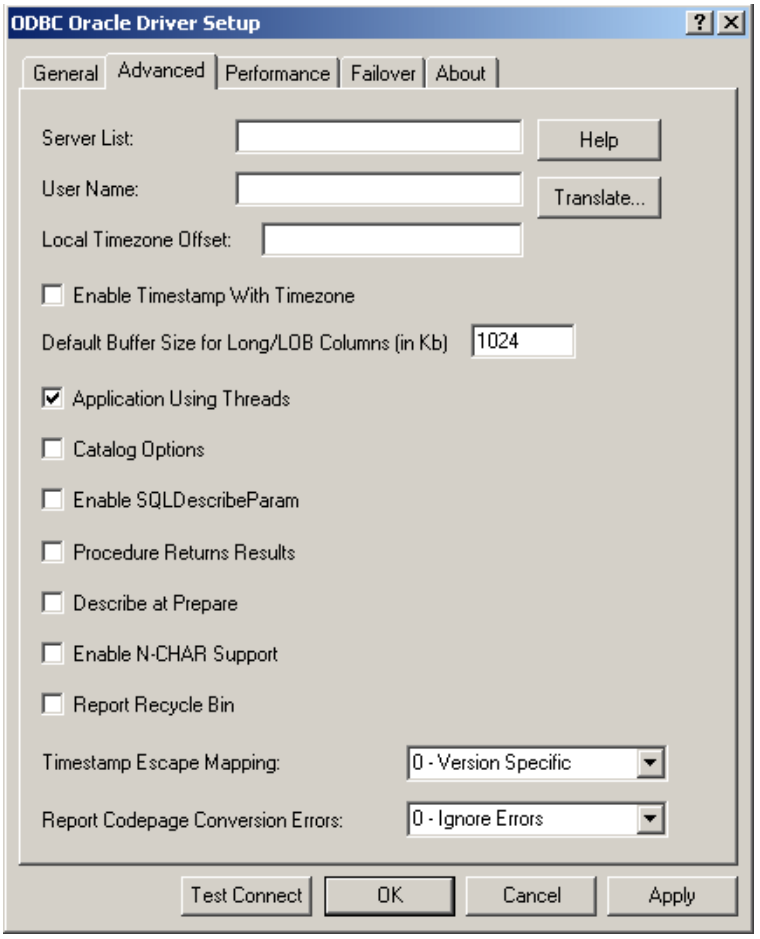


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 719)	None
Description (see page 721)	None
Server Name (see page 731)	None
Client Version (see page 717)	9iR1

4 Optionally, click the **Advanced** tab to specify additional data source settings.



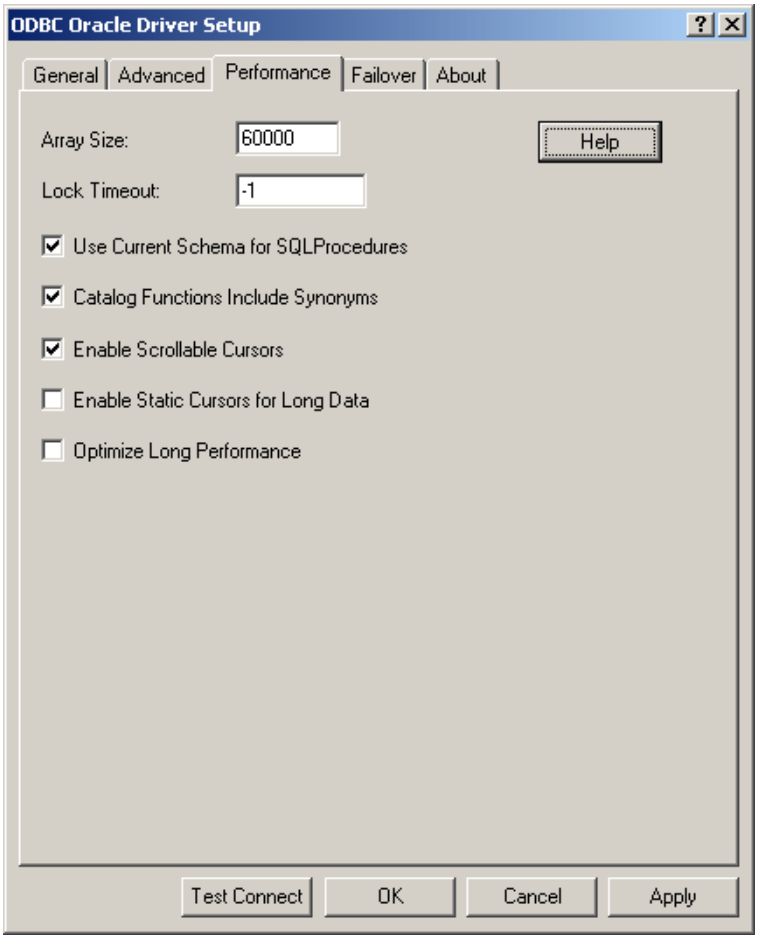
On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Server List (see page 730)	None
User Name (see page 733)	None
Local Timezone Offset (see page 726)	None
Enable Timestamp with Timezone (see page 724)	Disabled
Default Buffer Size for Long/LOB Columns (in Kb) (see page 719)	1024
Application Using Threads (see page 715)	Enabled
Catalog Options (see page 716)	Disabled
Enable SQLDescribeParam (see page 722)	Disabled
Procedure Returns Results (see page 728)	Disabled
Describe At Prepare (see page 720)	Disabled
Enable N-CHAR Support (see page 721)	Disabled
Report Recycle Bin (see page 730)	Disabled
Timestamp Escape Mapping (see page 732)	0 - Version Specific
Report Codepage Conversion Errors (see page 729)	0 - Ignore Errors
IANAAppCodePage (see page 724) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

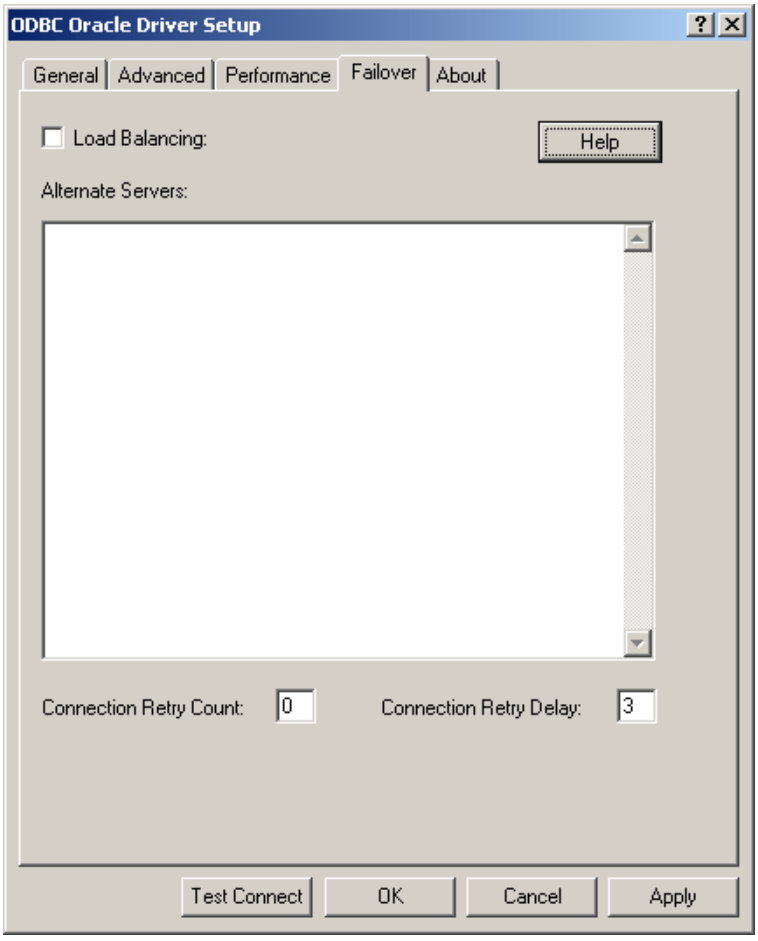
5 Optionally, click the **Performance** tab to specify performance data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Performance	Default
Array Size (see page 715)	60000
Lock Timeout (see page 726)	-1
Use Current Schema for SQLProcedures (see page 732)	Enabled
Catalog Functions Include Synonyms (see page 716)	Enabled
Enable Scrollable Cursors (see page 722)	Enabled
Enable Static Cursors for Long Data (see page 723)	Disabled
Optimize Long Performance (see page 727)	Disabled

6 Optionally, click the **Failover** tab to specify failover data source settings.



See "Using Failover" on page 85 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing (see page 725)	Disabled
Alternate Servers (see page 714)	None
Connection Retry Count (see page 718)	0
Connection Retry Delay (see page 718)	3

7 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 711 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

NOTE: If you are configuring alternate servers for use with the connection failover feature, be aware that the Test Connect button tests only the primary server, not the alternate servers.

8 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*= *value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 13-1 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;SRVR=QESRVR;UID=JOHN;PWD=XYZZY
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Oracle.dsn;SRVR=QESRVR;UID=JOHN;PWD=XYZZY
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect Oracle;SRVR=QESRVR;CV=10GR1;UID=JOHN;PWD=
XYZZY
```

If the server name contains a semicolon, enclose it in quotation marks:

```
DSN=Accounting;SRVR="QE;SRVR";UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.

In this dialog box, provide the following information:

1 In the Server Name field, type the client connection string of the computer containing the Oracle database tables you want to access. Or, select the string from the Server Name drop-down list, which displays the names you specified in the ODBC Oracle driver Setup dialog box.

   For local servers, use the SQL*Net connection string. If the SQL*Net connection string contains semicolons, enclose it in

quotation marks. Refer to your SQL*Net documentation for more information.

For remote servers, the Oracle TNS Client connection string is the alias name of the Oracle Listener on your network.

2 If required, type your Oracle user name.

3 If required, type your Oracle password.

4 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" on page 745 for details.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 13-1 lists the connection string attributes supported by the Oracle driver.

Table 13-1. Oracle Attribute Names

Attribute (Short Name)	Default
AlternateServers (ASVR)	None
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	60000
CatalogIncludesSynonyms (CIS)	1 (Enabled)
CatalogOptions (CO)	0 (Disabled)
ClientVersion (CV)	9iR1
ConnectionRetryCount (CRC)	0
ConnectionRetryDelay (CRD)	3
DataSourceName (DSN)	None
DefaultLongDataBuffLen (DLDBL)	1024
DescribeAtPrepare (DAP)	0 (Disabled)
Description (n/a)	Empty
EnableDescribeParam (EDP)	0 (Disabled)
EnableNcharSupport (ENS)	0 (Disabled)
EnableScrollableCursors (ESC)	1 (Enabled)
EnableStaticCursorsForLongData (ESCLD)	0 (Disabled)
EnableTimestampwithTimezone (ETWT)	0 (Disabled)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
LoadBalancing (LB)	0 (Disabled)
LocalTimezoneOffset (LTZO)	None
LockTimeout (LTO) (see page 726)	-1
LogonID (UID)	None
OptimizeLongPerformance (OLP)	0 (Disabled)
Password (PWD)	Empty
ProcedureRetResults (PRR)	0 (Disabled)
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
ReportRecycleBin (RRB)	0 (Disabled)

Table 13-1. Oracle Attribute Names (cont.)

Attribute (Short Name)	Default
ServerList	None
ServerName (SRVR)	None
TimestampEscapeMapping (TEM)	0 (Version Specific)
UseCurrentSchema (UCS)	1 (Enabled)

Alternate Servers

Attribute: AlternateServers (ASVR)

Description: A list of alternate database servers to which the driver will try 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: `(ServerName=`*`servervalue`*`[, . . .])`

You must specify the server name of each alternate server.

Example: The following Alternate Servers value defines two alternate database servers for connection failover:

```
(ServerName=AcctBackup1, ServerName=AcctBackup2)
```

Default: None

GUI tab: Failover tab on page 708

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI tab Advanced tab on page 704

Array Size

Attribute ArraySize (AS)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

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.

Default 60000

GUI Tab Performance tab on page 705

Catalog Functions Include Synonyms

Attribute CatalogIncludesSynonyms (CIS)

Description Determines whether synonyms are included in calls to SQLProcedures, SQLStatistics, and SQLProcedureColumns.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

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.

Default 1 (Enabled)

GUI Tab Performance tab on page 705

Catalog Options

Attribute CatalogOptions (CO)

Description Determines whether SQL_NULL_DATA is returned for the result columns REMARKS and COLUMN_DEF.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 704

Client Version

Attribute ClientVersion (CV)

Description A value to specify the Oracle client software version. The driver assumes that it is using the version of Oracle client software specified by this option to connect to an Oracle server.

Valid Values 8i | 9iR1 | 9iR2 | 10gR1

When set to 10gR1 and later, the driver binds all non-integer numerics as BINARY FLOAT and BINARY DOUBLE. When set to any Oracle version previous to Oracle10*g* R1, the driver binds non-integer numerics as if connected to an Oracle 9*i* R2 or earlier version of the server (regardless of the actual version of the server to which it is connected). When connecting to an Oracle 10*g* server with a pre-10*g* client, this attribute must be set to the same version as the actual Oracle client software in use; otherwise, numeric parameter bindings may fail. Versions of the Oracle client software prior to 10*g* R1 do not fully support the new features of the Oracle 10*g* database server.

Default 9iR1

GUI Tab General tab on page 703

Connection Retry Count

Attribute ConnectionRetryCount (CRC)

Description 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.

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 on page 708

Connection Retry Delay

Attribute ConnectionRetryDelay (CRD)

Description 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.

If set to 0, there is no delay between retries.

If set to *x*, the driver waits between connection retry attempts the specified number of seconds.

Default 3

GUI Tab Failover tab on page 708

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 703

Default Buffer Size for Long/LOB Columns (in Kb)

Attribute DefaultLongDataBuffLen (DLDBL)

Description The maximum length of data (in KB) the driver can fetch from Long/LOB columns in a single round trip and the maximum length of data that the driver can send using the SQL_DATA_AT_EXEC parameter.

NOTE: If this option is enabled, the Optimize Long Performance option is ignored.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values: An integer in multiples of 1024

The value must be in multiples of 1024 (for example, 1024, 2048). You need to increase the default value if the total size of any Long data exceeds 1 MB. This value is multiplied by 1024 to determine the total maximum length of fetched data. For example, if you enter a value of 2048, the maximum length of data would be 1024 x 2048, or 2097152 (2 MB).

Default: 1024

GUI Tab: Advanced tab on page 704

Describe At Prepare

Attribute: DescribeAtPrepare (DAP)

Description: Determines whether the driver describes the SQL statement at prepare time.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

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.

Default: 0 (Disabled)

GUI Tab: Advanced tab on page 704

Description

Attribute Description (n/a)

Description 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 on page 703

Enable N-CHAR Support

Attribute EnableNcharSupport (ENS)

Description 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.

See “Unicode Support” on page 740 for details.

NOTE: Valid only on Oracle 9*i* and higher.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 704

Enable Scrollable Cursors

Attribute EnableScrollableCursors (ESC)

Description Determines whether scrollable cursors, both Keyset and Static, are enabled for the data source.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

If set to 1 (Enabled), scrollable cursors are enabled for the data source.

If set to 0 (Disabled), scrollable cursors are not enabled.

Default 1 (Enabled)

GUI Tab Performance tab on page 705

Enable SQLDescribeParam

Attribute EnableDescribeParam (EDP)

Description Determines whether the SQLDescribeParam function describes all parameters with a data type of SQL_VARCHAR for Select statements. For Insert/Update/Delete statements and for stored procedures, the parameters are described as the actual Oracle data types on the Oracle server. This option must be enabled to access data when using Microsoft Remote Data Objects (RDO).

Valid Values 0 | 1

If set to 1 (Enabled), the SQLDescribeParam function describes all parameters with a data type of SQL_VARCHAR for Select statements.

If set to 0 (Disabled), the SQLDescribeParam function does not describe all parameters with a data type of SQL_VARCHAR for Select statements.

Default 0 (Disabled)

GUI Tab Advanced tab on page 704

Enable Static Cursors for Long Data

Attribute EnableStaticCursorsForLongData (ESCLD)

Description 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.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

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.

NOTE: You must enable this option if you want to persist a result set that contains Long data into an XML data file.

Default 0 (Disabled)

GUI Tab Performance tab on page 705

Enable Timestamp with Timezone

Attribute: EnableTimestampwithTimezone (ETWT)

Description: Determines whether the driver exposes timestamps with timezones to the application.

Valid Values: 0 | 1

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: 0 (Disabled)

GUI Tab: Advanced tab on page 704

IANAAppCodePage

Attribute: IANAAppCodePage (IACP)

Description: 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 704

Load Balancing

Attribute LoadBalancing (LB)

Description Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values 0 | 1

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).

NOTE: This option has no effect unless alternate servers are defined for the Alternate Servers connection option.

Default 0 (Disabled)

GUI Tab Failover tab on page 708

Local Timezone Offset

Attribute LocalTimezoneOffset (LTZO)

Description 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 on page 704

Lock Timeout

Attribute LockTimeout (LTO)

Description Specifies the amount of time, in seconds, the Oracle server waits for a lock to be released before generating an error when processing a Select...For Update statement on an Oracle 9*i* or higher server.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values -1 | 0 | *x*

where *x* is an integer that specifies a number of seconds.

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.

NOTE: If you are connected to an Oracle 8*i* server, any value greater than 0 is equivalent to the value -1.

Default -1

GUI Tab Performance tab on page 705

Optimize Long Performance

Attribute OptimizeLongPerformance (OLP)

Description Allows the driver to fetch Long data directly into the application's buffers rather than allocating buffers and making a copy. This option decreases fetch times on Long data; however, it can cause the application to be limited to one active statement per connection.

NOTE: If this option is enabled, the Default Buffer Size for Long/LOB Columns option is ignored.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver fetches Long data directly into the application's buffers rather than allocating buffers and making a copy.

If set to 0 (Disabled), the driver does not fetch Long data directly into the application's buffers.

Default 0 (Disabled)

GUI Tab Performance tab on page 705

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Procedure Returns Results

Attribute ProcedureRetResults (PRR)

Description Determines whether the driver returns result sets from stored procedures/functions.

See "MTS Support" on page 744 for details.

Valid Values 0 | 1

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.

Default 0 (Disabled)

GUI Tab Advanced tab on page 704

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 704

Report Recycle Bin

Attribute ReportRecycleBin (RRB)

Description Determines whether support is provided for reporting objects that are in the Oracle Recycle Bin.

On Oracle 10*g* R1 and higher, when a table is dropped, it is not actually removed from the database, but placed in the recycle bin instead.

Valid Values 0 | 1

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 on page 704

Server List

Attribute ServerList

Description A list of client connection strings that appear in the logon dialog box. This option applies to GUIs only and is not a runtime connection string attribute.

See "Performance Considerations" on page 733 for details about the logon dialog box.

Valid Values *string*

where *string* is a list of valid client connection strings. Separate the strings with commas. If the client connection string contains a comma, enclose it in quotation marks, for example, `"Serv,1", "Serv,2", "Serv,3"`.

Default Empty

GUI Tab Advanced tab on page 704

Server Name

Attribute ServerName (SRVR)

Description The client connection string of the computer containing the Oracle database tables you want to access.

Valid Values *string*

where *string* is a valid client connection string.

For local servers, use the SQL*Net connection string. If the SQL*Net connection string contains semicolons, enclose it in quotation marks. Refer to your SQL*Net documentation for more information.

For remote servers, the Oracle TNS Client connection string is the alias name of the Oracle Listener on your network.

Default None

GUI Tab General tab on page 703

Timestamp Escape Mapping

Attribute: TimestampEscapeMapping (TEM)

Description: Determines how the driver maps Date, Time, and Timestamp literals.

Valid Values: 0 | 1

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 on page 704

Use Current Schema for SQLProcedures

Attribute: UseCurrentSchema (UCS)

Description: Determines whether the driver returns only procedures owned by the current user when executing SQLProcedures.

This connection option can affect performance. See "Performance Considerations" on page 733 for details.

Valid Values: 0 | 1

When set to 1 (Enabled), the call for SQLProcedures is optimized, but only procedures owned by the user are returned.

When set to 0 (Disabled), the driver does not specify only the current user.

Default 1 (Enabled)

GUI Tab Performance tab on page 705

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.

You can also use OS Authentication to connect to your Oracle database. See "OS Authentication" on page 745 for details.

Valid Values *userid*

where *userid* is a valid user ID with permissions to access the database.

Default None

GUI Tab Advanced tab on page 704

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

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.

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 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.

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)

  NOTE: This option is not available with Oracle 8.

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.

Optimize Long Performance (OptimizeLongPerformance): When enabled, this option fetches Long data directly into the application's buffers rather than allocating buffers and making a copy. Also, when enabled, this option decreases fetch times on Long data; however, it can cause the application to be limited to one active statement per connection.

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).

Use Current Schema for SQLProcedures (UseCurrentSchema): If your application needs to access database objects owned only by the current user, performance of your application can be improved. In this case, the UseCurrentSchema attribute should be enabled (set to 1). When this attribute 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 attribute is equivalent to passing the Logon ID used on the connection as the SchemaName argument to the catalog functions.

Data Types

Table 13-2 shows how the Oracle data types are mapped to the standard ODBC data types. "Unicode Support" on page 740 lists Oracle to Unicode data type mappings.

Table 13-2. Oracle Data Types

Oracle	ODBC
BFILE [1]	SQL_LONGVARBINARY
BINARY DOUBLE [2]	SQL_REAL
BINARY FLOAT [2]	SQL_DOUBLE
BLOB [3]	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB [3]	SQL_LONGVARCHAR
DATE	SQL_TYPE_TIMESTAMP
LONG	SQL_LONGVARCHAR
LONG RAW	SQL_LONGVARBINARY
NUMBER	SQL_DOUBLE
NUMBER (p,s)	SQL_DECIMAL
RAW	SQL_VARBINARY
TIMESTAMP [4]	SQL_TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE [4]	SQL_TIMESTAMP
TIMESTAMP WITH TIMEZONE [4]	SQL_VARCHAR

***Table 13-2. Oracle Data Types** (cont.)*

VARCHAR2	SQL_VARCHAR
XMLType [5]	SQL_LONGVARCHAR

1. Read-Only
2. Supported only on Oracle 10*g* and higher.
3. Valid when connecting to Oracle 8 servers; these data types support output parameters to stored procedures
4. Supported only on Oracle 9*i* and higher.
5. Supported only on Oracle 9*i* R2 and higher.

The Oracle driver does not support any object types (also known as abstract data types). When the driver encounters an object type during data retrieval, it will return an Unknown Data Type error (SQL State HY000).

See "Retrieving Data Type Information" on page 77 for more information about data types.

XMLType

Oracle 9*i* R2 and higher supports the XMLType data type. The Oracle driver supports tables containing columns whose data type is specified as XMLType.

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.

Example

The following example from the Oracle Web site demonstrates how to create a table, insert data, and retrieve data. Creating a table with XMLType is done the same way as creating a regular table in the Oracle database, using standard SQL syntax:

```
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>
'));
```

Use the getClobVal function to retrieve the data:

```
SELECT p.podocument.getClobVal() FROM PURCHASEORDER p;
```

Unicode Support

The Oracle driver uses the NLS_LANG environment variable setting of the Oracle client to determine how to transmit data to the client.

On Windows, UNIX, and Linux, a Unicode setting is determined if the NLS_LANG environment variable is set to:

LANGUAGE_TERRITORY.CHARSET

where *CHARSET* is either UTF8, AL24UTFFSS, or AL32UTF8. For example:

AMERICAN_AMERICA.UTF8

Alternatively, on Windows, instead of the NLS_LANG environment variable, the value of the HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE*oracle_home_key* registry key can be set to:

LANGUAGE_TERRITORY.CHARSET

where *oracle_home_key* is HOME0 for Oracle 9*i* R2 and earlier, and is the Oracle home name used at the time of client installation for Oracle 10*g*.

If the *CHARSET* is a Unicode setting and a Unicode application is accessing the driver, then no data conversion is necessary. If an ANSI application is accessing the driver, then the driver must convert the data from the application from ANSI to Unicode (UTF-8) for the client.

If the *CHARSET* is ANSI and an ANSI application is accessing the driver, then no data conversion is necessary. If a Unicode application is accessing the driver, then the driver must convert the data from the application from Unicode to ANSI for the client.

If NLS_LANG is set to UTF-8, the Oracle driver maps the Oracle data types to Unicode data types as shown in the following table:

Oracle Data Type	Mapped to. . .
CHAR	SQL_WCHAR
CLOB	SQL_WLONGVARCHAR
VARCHAR2	SQL_WVARCHAR
LONG	SQL_WLONGVARCHAR

The driver also continues to map these Oracle data types to the normal character data types. See "Data Types" on page 737 for these mappings.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Unexpected Characters

Users are sometimes surprised when they insert a character into a database, only to have a different character displayed when they fetch it from the database. There are many reasons this can happen, but it most often involves code page issues, not driver errors.

Client and server machines in a database system each use code pages, which can be identified by a name or a number, such as Shift_JIS (Japanese) or cp1252 (Windows English). A code page is a mapping that associates a sequence of bits, called a code point, with a specific character. Code pages include the characters and symbols of one or more languages. Regardless of geographical location, a machine can be configured to use a specific code page. Most of the time, a client and database server would use similar, if not identical, code pages. For example, a client and server might use two different Japanese code pages, such as Shift_JIS and EUC_JP, but they would still share many Japanese characters in common. These characters might, however, be represented by different code points in each code page. This introduces the need to convert between code pages to maintain data integrity. In some cases, no one-to-one character correspondence exists between the two code points. This causes a substitution character to be used, which can result in displaying an unexpected character on a fetch.

When the driver on the client machine opens a connection with the database server, the driver determines the code pages being used on the client and the server. This is determined from the

Active Code Page on a Windows-based machine. If the client machine is UNIX-based, the driver checks the IANAAppCodePage option (see "IANAAppCodePage" on page 724). 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.

MTS Support

On Windows, the driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC) using the XA Protocol. For a general discussion of MTS and DTC, refer to the help file of the Microsoft Transaction Server SDK.

NOTE: The DataDirect Connect *for* ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 *for* ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

To enable DTC support, you must be accessing Oracle 8.0.5 or higher servers using Oracle Net8 Client 8.1.6 or higher.

OS Authentication

On Windows, UNIX, and Linux, 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.

Support for Oracle RAC

Oracle introduced Real Application Clusters (RAC) with Oracle 9*i*, and RAC is also a key feature of Oracle 10*g*. 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 a DataDirect Connect Series *for* ODBC driver. There is no additional configuration required.

C*onnection 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 "Using Failover" on page 85 for details about how these features work in DataDirect Connect Series *for* ODBC drivers.

Support of Materialized Views

When connected to an Oracle 9*i* or higher server, the Oracle driver supports the creation of materialized views. Materialized views are like any other database view with the following additions: the results are stored as a database object and the results can be updated on a schedule determined by the Create View statement.

Materialized views improve performance for data warehousing and replication. Refer to the Oracle documentation for more information about materialized views.

Stored Procedure Results

When you enable the Procedure Returns Results connection option, the driver returns 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 requires that stored procedures be in a certain format. First, a package must be created to define all of the cursors used in the procedure; then, the procedure can be created using the new cursor. 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.

Configuring Connection Failover

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

NOTE: If you are persisting a result set that contains Long data, you must enable the EnableStaticCursorsforLongData connection string attribute.

Isolation and Lock Levels Supported

Oracle supports isolation level 1 (read committed) and isolation level 3 (serializable). Oracle supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The Oracle driver also supports the following functions:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLProcedureColumns
- SQLSetPos
- SQLTablePrivileges

The driver supports the core SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Oracle driver.

Number of Connections and Statements Supported

The Oracle driver supports multiple connections and multiple statements per connection.

Using Arrays of Parameters

Oracle 8*i* and higher databases natively support parameter arrays, and the Oracle driver, in turn, supports them when connected to these versions of Oracle databases. When designing an application for performance, using native parameter arrays for bulk inserts or updates, for example, can improve performance. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

If the database does not support parameter arrays, the Oracle driver emulates them so that you can design your applications to use arrays of parameters and take advantage of the performance improvements where applicable. The driver emulates parameter arrays by sending individual rows to the database.

14 The SQL Server Classic Wire Protocol Driver

The DataDirect Connect Series *for* ODBC SQL Server Classic Wire Protocol driver (the SQL Server Classic Wire Protocol driver) is available in both 32- and 64-bit versions and supports:

- Microsoft SQL Server 2008
- Microsoft SQL Server 2005
- Microsoft SQL Server 2000 Service Packs 1, 2, 3, 3a, and 4
- Microsoft SQL Server 2000 Desktop Engine (MSDE 2000)
- Microsoft SQL Server 2000 Enterprise Edition (64-bit)
- Microsoft SQL Server 7.0

The SQL Server Classic Wire Protocol driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the SQL Server Classic Wire Protocol driver.

Driver Requirements

The driver has no client requirements.

Windows

For support of Microsoft SQL Server 7.0, 2000, and 2005, the driver requires the SQL Server 7.0 versions of Net-Library DLL files, which are installed when you install the SQL Server Classic Wire Protocol driver. The driver communicates with network software through the SQL Server Net-Library interface.

UNIX and Linux

To use the SQL Server Classic Wire Protocol driver on UNIX and Linux, you must have TCP/IP configured on both the UNIX and Linux clients and the Windows server on which the Microsoft SQL Server database resides. The UNIX and Linux SQL Server TCP/IP network client library is built into the SQL Server Classic Wire Protocol driver on UNIX and Linux.

The Microsoft SQL Server Client configuration has been merged with the ODBC driver configuration and is set in the system information file.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 “Quick Start Connect” on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 759, Table 14-1 on page 763, and Table 14-2 on page 765 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Each of Table 14-1 on page 763 and Table 14-2 on page 765 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Microsoft SQL Server data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

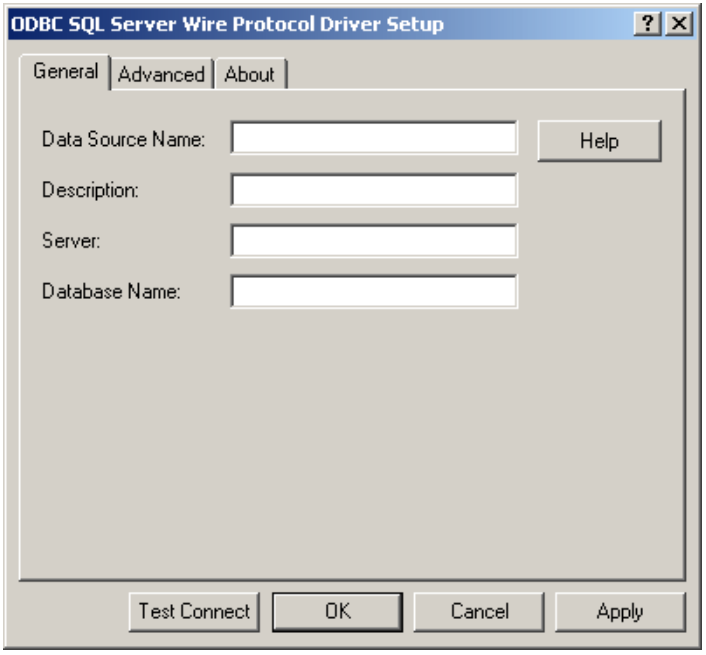
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

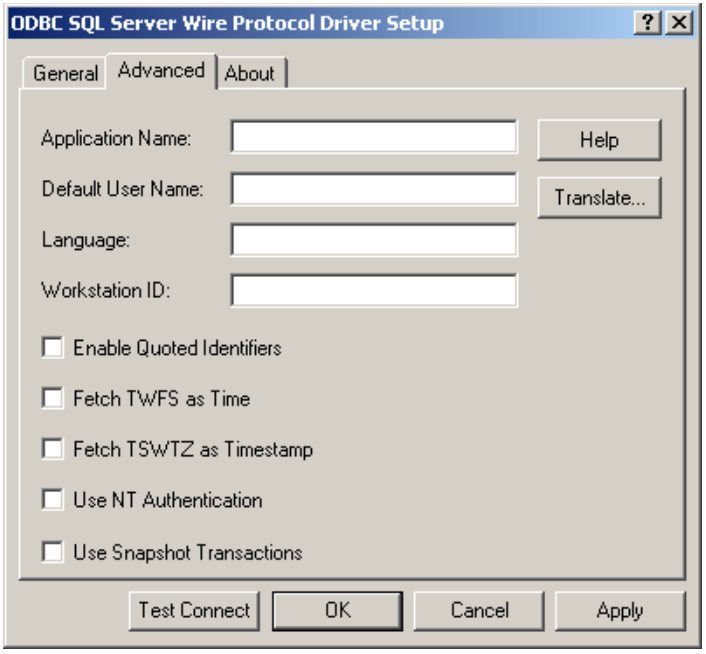


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 770)	None
Description (see page 771)	None
SERVER (see page 779)	None
Database Name (see page 770)	None

4 Optionally, click the **Advanced** tab to specify additional data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Application Name (see page 767)	None
Default User Name (see page 771)	None
Language (see page 775)	None
Workstation ID (see page 783)	None
Enable Quoted Identifiers (see page 772)	Disabled
Fetch TWFS as Time (see page 773)	Disabled
Fetch TSWTZ as Timestamp (see page 772)	Disabled

Connection Options: Advanced	Default
Use NT Authentication (see page 781) WINDOWS ONLY	Disabled
Use Snapshot Transactions (see page 782)	Disabled
IANAAppCodePage (see page 774) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

5 Optionally, click the **Failover** tab to specify failover data source settings. This tab is available only on UNIX and Linux.

See "Using Failover" on page 85 for a general description of failover and its related connection options.

On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Failover	Default
Load Balancing (see page 775)	Disabled
Alternate Servers (see page 766)	None
Connection Retry Count (see page 769)	0
Connection Retry Delay (see page 770)	3

6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A Login dialog box appears; see "Using a Connection String" on page 759 for details. Note that the information you enter in the Login 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**.

7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 14-1 and Table 14-2 list 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 Microsoft SQL Server is:

```
DSN=ACCOUNTING;DATABASE=ACCT
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=SQLServer.dsn;DATABASE=ACCT
```

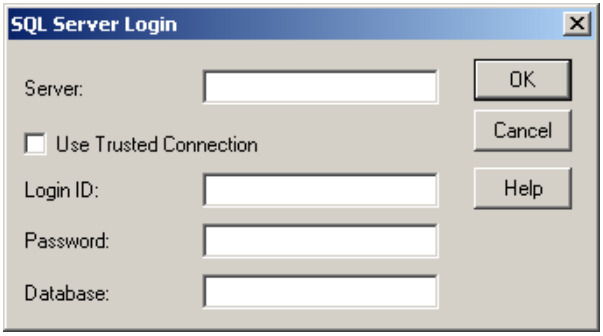
A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect SQL Server Classic Wire Protocol;
DATABASE=ACCT;SERVER=SQL2;UID=JOHN;PWD=XYZZY
```

The connection string attribute names are case-sensitive.

Using a Login Dialog Box

Some ODBC applications display a Login dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.



In the Login dialog box, provide the following information:

1 Type an IP address in Server in the following format: *IP_address*, *port_number*. For example, you can enter `199.226.224.34,5000`. If your network supports named servers, you can specify an address as: *server_name*, *port_number*. For example, you can enter `SSserver,5000`.

The IP address can be specified in IPv4 on Windows, and in either IPv4 or IPv6 format, or a combination of the two, on UNIX. See "Using IP Addresses" on page 70 for details about these formats.

To specify a named instance of Microsoft SQL Server, use the format: *server_name**instance_name*. If only a server name is specified with no instance name, the driver uses the default named instance on the server.

Type the name of a server on your network. It must be an entry on the Alias tab of the SQL Server Network Client Utility or the network name of a server running Microsoft SQL Server.

You can enter `(local)` when the driver is on the same computer as the Microsoft SQL Server database. You can connect to a local copy of Microsoft SQL Server, even when it is a non-networked version. Microsoft SQL Server 2000 and higher support multiple instances of Microsoft SQL Server running on the same computer.

2 Select the **Use Trusted Connection** check box to specify that the SQL Server Classic Wire Protocol driver request a secure (or trusted) connection to Microsoft SQL Server running on Windows NT, Windows 2000, Windows XP, or Windows Server 2003. SQL Server uses integrated login security to establish connections using this data source, regardless of the current login security mode at the server. Any login ID or password supplied is ignored. The Microsoft SQL Server system administrator must have associated your Windows network ID with a Microsoft SQL Server login ID.

   Clear this box to specify that Microsoft SQL Server use standard login security to establish connections using this data source. You must specify a login ID and password for all connection requests.

3 Type the Microsoft SQL Server login ID to use for the connection if Use Trusted Connection is not selected. If Use Trusted Connection is selected, the Login ID field is disabled.

4 Type the password to use for the connection if Use Trusted Connection is not selected. If Use Trusted Connection is selected, the Password field is disabled.

5 Type the name of the database to which you want to connect. If you do not specify a value, the default database defined by Microsoft SQL Server is used.

6 Click **OK** to log on to the Microsoft SQL Server database installed on the server you specified and to update the values in the Registry.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

NOTE: SQL Server driver connection string attributes do not use short name equivalents.

Table 14-1 lists the connection string attributes supported by the SQL Server Classic Wire Protocol driver on Windows.

Windows

Table 14-1. SQL Server Classic Wire Protocol Attribute Names on Windows

Attribute	Default
AnsiNPW	yes (Enabled)
APP	None
AttachDBFileName	None

Table 14-1. SQL Server Classic Wire Protocol Attribute Names on Windows *(cont.)*

Attribute	Default
AutoTranslate	yes (Enabled)
DATABASE	None
DataSourceName (DSN)	None
Description (n/a)	None
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
LANGUAGE	None
Network	None
PWD (use Password for odbc.ini file)	None
QueryLog_On	no (Disabled)
QueryLogFile	None
QueryLogTime	None
QuotedID	no (Disabled)
Regional	yes (Enabled)
SAVEFILE	None
SERVER	None
SnapshotSerializable	0 (Disabled)
StatsLog_On	no (Disabled)
StatsLogFile	None
Trusted_Connection	no (Disabled)
UID (use LogonID for odbc.ini file)	None
WSID	None

UNIX and Linux

Table 14-2. lists the connection string attributes supported by the SQL Server Classic Wire Protocol driver on UNIX/Linux.

NOTE: SQL Server driver connection string attributes do not use short name equivalents.

Table 14-2. SQL Server Classic Wire Protocol Attribute Names on UNIX/Linux

Attribute	Default
Address	None
AlternateServers	None
AnsiNPW	yes (Enabled)
APP	None
ConnectionRetryCount	0
ConnectionRetryDelay	3
Data Source Name	None
DATABASE	None
Description (n/a)	None
FetchTSWTZasTimestamp (FTSWTZAT)	0 (Disabled)
FetchTWFSasTime (FTWFSAT)	0 (Disabled)
IANAAppCodePage	4 (ISO 8559-1 Latin-1)
LANGUAGE	None
LoadBalancing (LB)	0 (Disabled
PWD (use Password for odbc.ini file)	None
QuotedID	no (Disabled)
SnapshotSerializable	0 (Disabled)
UID (use LogonID for odbc.ini file)	None
WSID	None

Alternate Servers

Attribute AlternateServers

Description A list of alternate database servers to which the driver will try 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 `(Address=`*`addressvalue`*`[, . . .])`

You must specify the network address of each alternate server.

NOTE: An alternate server address in IPv6 format must be enclosed in double quotation marks.

Example The following two Alternate Servers values define two alternate database servers for connection failover:

```
AlternateServers=(Address=\\MySQLServer\Instance1,
Address="255.125.1.11, 5002")
```

Default None

GUI Tab Failover tab on page 758

AnsiNPW

Attribute AnsiNPW

Description Determines whether ANSI-defined behaviors are exposed.

Valid Values yes | no

When set to yes (Enabled), the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation. If the driver appears to be truncating trailing blank spaces, this attribute should be set to no.

When set to no (Disabled), ANSI-defined behaviors are not exposed.

Default yes (Enabled)

GUI Tab n/a

Application Name

Attribute APP

Description The name Microsoft SQL Server uses to identify your application.

Valid Values *string*

where *string* is your application name.

Default None

GUI Tab Advanced tab on page 757

AttachDBFileName

Attribute AttachDBFileName

Description The name of the primary file of an attachable database.

Valid Values *string*

where *string* is name of the primary file of an attachable database.

Include the full path and escape any slash (\ ) characters if using a C character string variable:

AttachDBFileName=C:\\MyFolder\\MyDB.mdf

This database is attached and becomes the default database for the connection. To use AttachDBFileName, you must also specify the database name in either the SQLDriverConnnect DATABASE parameter or the SQL_COPT_CURRENT_CATALOG connection

attribute. If the database was previously attached, Microsoft SQL Server will not reattach it; it will use the attached database as the default for the connection.

Default None

GUI Tab n/a

AutoTranslate

Attribute AutoTranslate

Description Determines how ANSI character strings are translated.

Valid Values yes | no

If set to yes (Enabled), ANSI character strings sent between the client and server are translated by converting through Unicode to minimize problems in matching extended characters between the code pages on the client and the server.

These conversions are performed on the client by the SQL Server Classic Wire Protocol driver. This requires that the same ANSI code page (ACP) used on the server be available on the client.

These settings have no effect on the conversions that occur for the following transfers:

- Unicode SQL_C_WCHAR client data sent to char, varchar, or text on the server.
- Char, varchar, or text server data sent to a Unicode SQL_C_WCHAR variable on the client.
- ANSI SQL_C_CHAR client data sent to Unicode nchar, nvarchar, or ntext on the server.
- Unicode char, varchar, or text server data sent to an ANSI SQL_C_CHAR variable on the client.

If set to no (Disabled), character translation is not performed.

The SQL Server Classic Wire Protocol driver does not translate client ANSI character SQL_C_CHAR data sent to char, varchar, or text variables, parameters, or columns on the server. No translation is performed on char, varchar, or text data sent from the server to SQL_C_CHAR variables on the client. If the client and Microsoft SQL Server are using different ACPs, then extended characters can be misinterpreted.

Default yes (Enabled)

GUI Tab n/a

Connection Retry Count

Attribute ConnectionRetryCount

Description 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.

The Connection Retry Delay option specifies the wait interval, in seconds, to occur between retry attempts.

Valid Values 0 | *x*

where *x* is a positive integer from 1 to 65535.

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 on page 757

Connection Retry Delay

Attribute: ConnectionRetryDelay

Description: The number of seconds the driver waits between connection retry attempts when Connection Retry Count is set to a positive integer.

Valid Values: 0 | *x*

where *x* is a positive integer from 1 to 65535.

If set to 0, there is no delay between retries.

If set to *x*, the driver waits between connection retry attempts the specified number of seconds.

Default: 3

GUI Tab: Failover tab on page 757

Data Source Name

Attribute: DataSourceName (DSN)

Description: 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 on page 756

Database Name

Attribute: DATABASE

Description: The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 756

Default User Name

Attribute UID (use LogonID for odbc.ini file)

Description The default user ID used to connect to your database. Your ODBC application may override this value or you may override it in the logon dialog box or connection string.

Valid Values *userid*

where *userid* is a valid user ID with permissions to access the database.

Default None

GUI Tab Advanced tab on page 757

Description

Attribute Description (n/a)

Description 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 on page 756

Enable Quoted Identifiers

Attribute QuotedID

Description Determines whether the driver allows the use of quoted identifiers.

Valid Values yes | no

If set to yes (Enabled), Microsoft SQL Server enforces ANSI rules regarding quotation marks. Double quotation marks can only be used for identifiers, such as column and table names. Character strings must be enclosed in single quotation marks, for example:

```
SELECT "au_id"
FROM "authors"
WHERE "au_lname" = 'O''Brien'
```

If set to no (Disabled), applications that use quoted identifiers encounter errors when they generate SQL statements with quoted identifiers.

Default no (Disabled)

GUI Tab Advanced tab on page 757

Fetch TSWTZ as Timestamp

Attribute FetchTSWTZasTimestamp (FTSWTZAT)

Description Determines whether the driver returns column values with the timestamp with time zone data type as the ODBC data type SQL_TYPE_TIMESTAMP or SQL_VARCHAR.

Valid Values 0 | 1

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 timezone 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 timezone information in the fetched value.

Default 0 (Disabled)

GUI Tab Advanced tab on page 757

Fetch TWFS as Time

Attribute FetchTWFSasTime (FTWFSAT)

Description Determines whether the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME or SQL_TYPE_TIMESTAMP.

Supported only for Microsoft SQL Server 2008.

Valid Values 0 | 1

If set to 1 (Enabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIME. The fractional seconds portion of the value is truncated.

If set to 0 (Disabled), the driver returns column values with the time data type as the ODBC data type SQL_TYPE_TIMESTAMP. The fractional seconds portion of the value is preserved. Time columns are not searchable when they are described and fetched as timestamp

NOTE: When returning time with fractional seconds data as SQL_TYPE_TIMESTAMP, the Year, Month and Day parts of the timestamp must be set to zero.

Default 0 (Disabled)

GUI Tab Advanced tab on page 757

IANAAppCodePage

Attribute IANAAppCodePage

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 757

Language

Attribute LANGUAGE

Description The national language to use for Microsoft SQL Server system messages.

Valid Values *lang*

where *lang* is the language to use for Microsoft SQL Server system messages. This overrides the default language specified for the login on the server. If no language is specified, the connection uses the default language specified for the login on the server.

Default None

GUI Tab Advanced tab on page 757

Load Balancing

Attribute LoadBalancing (LB)

Description Determines whether the driver uses client load balancing in its attempts to connect to the database servers (primary and alternate). You can specify one or multiple alternate servers by setting the Alternate Servers option.

Valid Values 0 | 1

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 on page 758

Network

Attribute Network

Description The name of a network library dynamic-link library.

Valid Values *string*

where *string* is the name of a network library dynamic-link library. The name need not include the path and must not include the .DLL file name extension, for example, `Network=dbnmpntw`.

Default None

GUI Tab n/a

PWD

Attribute PWD (use Password for odbc.ini file)

Description 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.

PWD need not be specified if the login has a NULL password.

Default None

GUI Tab n/a

QueryLogFile

Attribute QueryLogFile

Description The full path and file name of a file to be used for logging data about long-running queries. The QueryLog_On option must be set to yes.

Valid Values *string*

where *string* is the full path and file name of the file to be used for logging data.

Default None

GUI Tab n/a

QueryLog_On

Attribute QueryLog_On

Description Determines whether data about long-running queries data is logged.

Valid Values yes | no

When set to yes (Enabled), logging data about long-running queries data is enabled on the connection.

When set to no (Disabled), long-running query data is not logged.

Default no (Disabled)

GUI Tab n/a

QueryLogTime

Attribute QueryLogTime

Description A digit character string specifying the threshold for logging data about long-running queries.

Valid Values *string*

where *string* is a digit character string specifying the threshold in milliseconds, for logging data.

Any query that does not receive a response in the time specified is written to the long-running query log file.

Default None

GUI Tab n/a

Regional

Attribute Regional

Description Determines how currency, date, and time data are converted.

Valid Values yes | no

When set to yes (Enabled), the SQL Server Classic Wire Protocol driver uses client settings when converting currency, date, and time data to character data. The conversion is one way only; the driver does not recognize non-ODBC standard formats for date strings or currency values.

When set to no (Disabled), the driver uses ODBC standard strings to represent currency, date, and time data that is converted to string data.

Default yes (Enabled)

GUI Tab n/a

SAVEFILE

Attribute SAVEFILE

Description The name of an ODBC data source file into which the attributes of the current connection are saved.

Valid Values *string*

where *string* is the name of an ODBC data source file into which the attributes of the current connection are saved if the connection is successful.

Default None

GUI Tab n/a

SERVER

Attribute SERVER

Attribute Address

Description The location of the server.

Valid Values *IP_address* | *named_server* | *named_instance* | *server_name*

where

IP_address is the IP address of the server to which you want to connect. Specify this address as: *IP_address*, *port_number*. For example, you can enter `199.226.224.34, 5000`.

The IP address can be specified in either IPv4 or IPv6 format, or a combination of the two. See "Using IP Addresses" on page 70 for details about these formats.

named_server is the named server address of the server to which you want to connect. Specify this address as: *named_server*, *port_number*. For example, you can enter `SSserver, 5000`.

named_instance is a named instance of Microsoft SQL Server. Specify this address as: *server_name**instance_name*. If only a server name is specified with no instance name, the driver uses the default named instance on the server.

server_name is the name of a server on your network. It must be an entry on the Alias tab of the SQL Server Network Client Utility or the network name of a server running Microsoft SQL Server. You can enter (local) when the driver is on the same computer as the Microsoft SQL Server database. You can connect to a local copy of Microsoft SQL Server, even when it is a non-networked version. Microsoft SQL Server 2000 and higher support multiple instances of Microsoft SQL Server running on the same computer.

Default None

GUI Tab General tab on page 756

StatsLogFile

Attribute StatsLogFile

Description The full path and file name of a file to be used for recording SQL Server Classic Wire Protocol driver performance data. The StatsLog_On option must be set to yes.

Valid Values *string*

where *string* is the full path and file name of the file to be used for recording data.

Default None

GUI Tab n/a

StatsLog_On

Attribute StatsLog_On

Description Determines whether SQL Server Classic Wire Protocol driver performance data is made available.

Valid Values yes | no

When set to yes (Enabled), SQL Server Classic Wire Protocol driver performance data is captured.

When set to no (Disabled), SQL Server Classic Wire Protocol driver performance data is not available on the connection.

Default no (Disabled)

GUI Tab n/a

Use NT Authentication

Attribute Trusted_Connection

Description Specifies that the SQL Server Classic Wire Protocol driver request a secure (or trusted) connection to Microsoft SQL Server running on Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

Valid Values 0 | 1

When set to 1 (Enabled), Microsoft SQL Server uses integrated login security to establish connections using this data source, regardless of the current login security mode at the server. Any login ID or password supplied is ignored. The Microsoft SQL Server system administrator must have associated your Windows network ID with a Microsoft SQL Server login ID.

When set to 0 (Disabled), Microsoft SQL Server uses standard login security to establish connections using this data source. In this case, you must specify a login ID and password for all connection requests.

Default 0 (Disabled)

GUI Tab Advanced tab on page 757

Use Snapshot Transactions

Attribute SnapshotSerializable

Description Allows your application to use the snapshot isolation level if your Microsoft SQL Server database is configured for Snapshot isolation. Supported only for Microsoft SQL Server 2005 and higher.

See "Isolation and Lock Levels Supported" on page 787 for details about using the snapshot isolation level.

This connection option can affect performance. See "Performance Considerations" on page 783 for details.

Valid Values 0 | 1

When set to 1 (Enabled) and your application has the transaction isolation level set to serializable, the application uses the snapshot isolation level.

When set to 0 (Disabled) and your application has the transaction isolation level set to serializable, the application uses the serializable isolation level.

This option is useful for existing applications that set the isolation level to serializable. Use Snapshot Transactions in this case allows you to change to the snapshot isolation level with no or minimum code changes. If developing a new application, you can code it to set the connection attribute SQL_COPT_SS_TXN_ISOLATION to the value SQL_TXN_SS_SNAPSHOT.

Default 0 (Disabled)

GUI Tab Advanced tab on page 757

Workstation ID

Attribute	WSID
Description	The workstation ID that is used by the client.
Valid Values	*string* where *string* is the workstation ID.
Default	None
GUI Tab	Advanced tab on page 757

Performance Considerations

Use Snapshot Transactions (SnapshotSerializable): You must have your Microsoft SQL Server 2005 and higher database configured for snapshot isolation for this connection option to work. Snapshot Isolation provides transaction-level read consistency and an optimistic approach to data modifications by not acquiring locks on data until data is to be modified. This Microsoft SQL Server 2005 and higher feature can be useful if you want to consistently return the same result set even if another transaction has changed the data and 1) your application executes many read operations or 2) your application has long running transactions that could potentially block users from reading data. This feature has the potential to eliminate data contention between read operations and update operations. When this connection option is enabled, performance is improved due to increased concurrency.

See "Using The Snapshot Isolation Level" on page 788 for details.

Data Types

Table 14-3 shows how the Microsoft SQL Server data types are mapped to the standard ODBC data types. "Unicode Support" on page 786 lists Microsoft SQL Server to Unicode data type mappings.

Table 14-3. Microsoft SQL Server Data Types

SQL Server	ODBC
binary	SQL_BINARY
bigint [1]	SQL_BIGINT
bit	SQL_BIT
char	SQL_CHAR
date [2]	SQL_TYPE_DATE
datetime	SQL_TYPE_TIMESTAMP
datetime2 [2]	SQL_TYPE_TIMESTAMP
datetimeoffset [2] [3]	SQL_WVARCHAR
decimal	SQL_DECIMAL
decimal() identity	SQL_DECIMAL
float	SQL_FLOAT
image	SQL_LONGVARBINARY
int	SQL_INTEGER
int identity	SQL_INTEGER
money	SQL_DECIMAL
numeric	SQL_NUMERIC
numeric() identity	SQL_NUMERIC
real	SQL_REAL
smalldatetime	SQL_TYPE_TIMESTAMP
smallint	SQL_SMALLINT
smallint identity	SQL_SMALLINT
smallmoney	SQL_DECIMAL

Table 14-3. Microsoft SQL Server Data Types (cont.)

SQL Server	ODBC
text	SQL_LONGVARCHAR
time [2] [4]	SQL_TYPE_TIMESTAMP
timestamp	SQL_VARBINARY
tinyint	SQL_TINYINT
tinyint identity	SQL_TINYINT
uniqueidentifier	SQL_GUID
varbinary	SQL_VARBINARY
varbinary(max) [5]	SQL_LONGVARBINARY
varchar	SQL_VARCHAR
varchar(max) [5]	SQL_LONGVARCHAR

1. Bigint supported on Windows driver only.
2. Supported only on Microsoft SQL Server 2008 and higher.
3. Datetimeoffset mapping changes based on the setting of the Fetch TSWTZ as Timestamp option.
4. Time mapping changes based on the setting of the Fetch TWFS as Time option.
5. Supported only on Microsoft SQL Server 2005 and higher.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Unicode Support

The SQL Server Classic Wire Protocol driver maps the Microsoft SQL Server data types to Unicode data types as shown in the following table:

SQL Server Data Type	Mapped to. . .
nchar	SQL_WCHAR
ntext	SQL_WLONGVARCHAR
nvarchar	SQL_WVARCHAR
nvarchar(max) [1]	SQL_WLONGVARCHAR
sysname	SQL_WVARCHAR
xml [1]	SQL_WLONGVARCHAR

1. Supported only for Microsoft SQL Server 2005 and higher.

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Configuring Connection Failover

Only the SQL Server Classic Wire Protocol driver on UNIX and Linux supports this feature.

The driver supports failover and its related connection options. See "Using Failover" on page 85 for a general description of failover and its implementation, as well as examples.

Isolation and Lock Levels Supported

Microsoft SQL Server supports isolation levels 0 (Read Uncommitted), 1 (Read Committed), 2 (Repeatable Read), and 3 (Serializable). Microsoft SQL Server supports row-level and table-level locking. Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

Microsoft SQL Server 2005 and higher supports the following additional isolation levels:

- Snapshot
- Read Committed with Snapshots
- Read Committed with Locks (equivalent to Read Committed in previous Microsoft SQL Server versions)

Using The Snapshot Isolation Level

The Snapshot isolation level is available only with Microsoft SQL Server 2005 and higher. Setting the SnapshotSerializable connection string attribute changes the behavior of the Serializable isolation level to use the Snapshot Isolation level. This allows an application to use the Snapshot Isolation level with minimal or no code changes.

If you are writing a new application, you may want to code it to set the connection attribute SQL_COPT_SS_TXN_ISOLATION to the value SQL_TXN_SS_SNAPSHOT. The application then uses the snapshot isolation level without requiring the Use Snapshot Transactions connection option.

See "Use Snapshot Transactions" on page 782 for additional information.

ODBC Conformance Level

The SQL Server Classic Wire Protocol driver supports ODBC conformance level 2.

The following function is supported: SQLDescribeParam.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the SQL Server Classic Wire Protocol driver.

Number of Connections and Statements Supported

The SQL Server Classic Wire Protocol driver supports multiple connections and a single statement per connection.

Using Arrays of Parameters

Microsoft SQL Server databases natively support parameter arrays, and the SQL Server Classic 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. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* for more information about using arrays of parameters to improve performance.

15 The Driver for the Teradata Database

The DataDirect Connect Series *for* ODBC driver for the Teradata database is available in both 32- and 64-bit versions and supports the following Teradata database servers when using the appropriate client software:

- 12.0
- V2R6.2
- V2R6.1
- V2R6.0
- V2R5.1.*x*

The driver is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the driver.

Driver Requirements

The driver requires Teradata Tools and Utilities (TTU) 8.2 or higher, which includes CLIv2, TGSS, and ICU client software, on all platforms. It requires TTU 12.0 to support 12.0 functionality.

NOTE: TTU 12.0 is not available for the Itanium II platform. You can use TTU 8.2 on an Itanium II client to connect to a Teradata 12.0 database, but functionality is limited to that of TTU 8.2.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 802 and Table 15-1 on page 806 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and

storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 15-1 on page 806 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete

description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Teradata data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

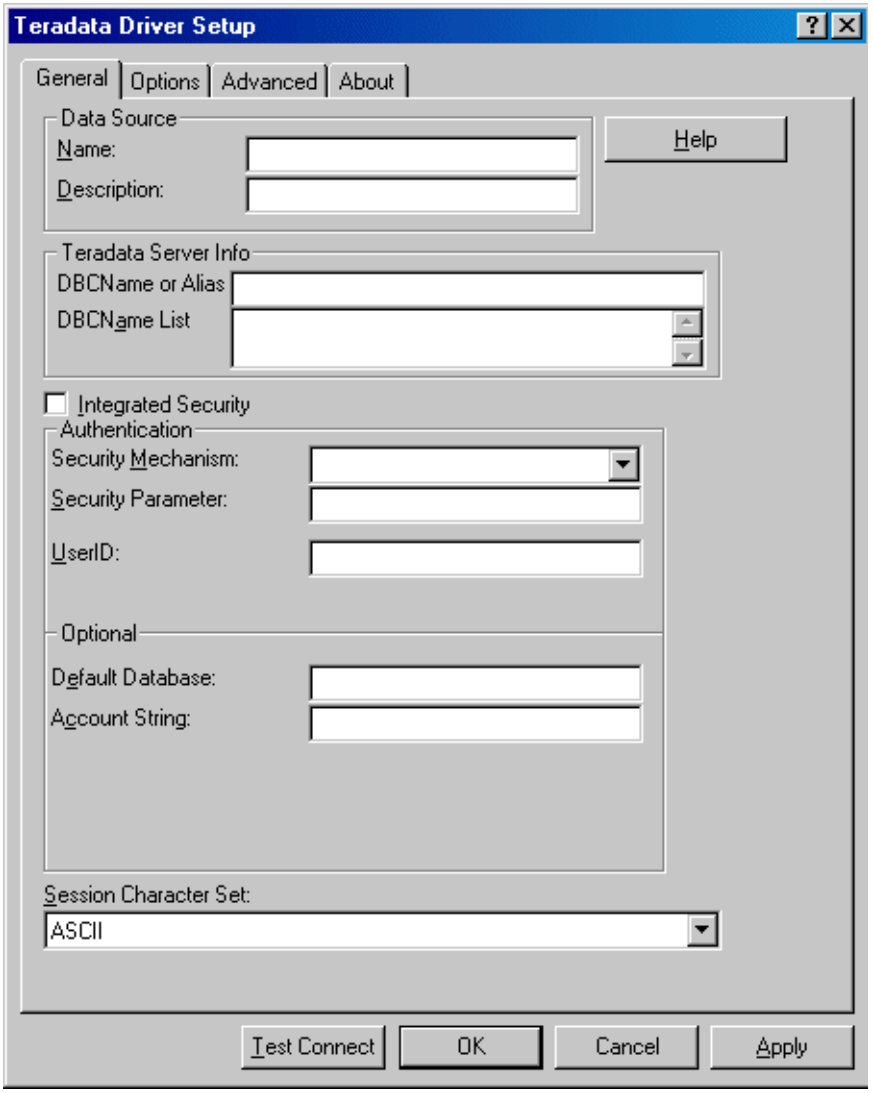
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

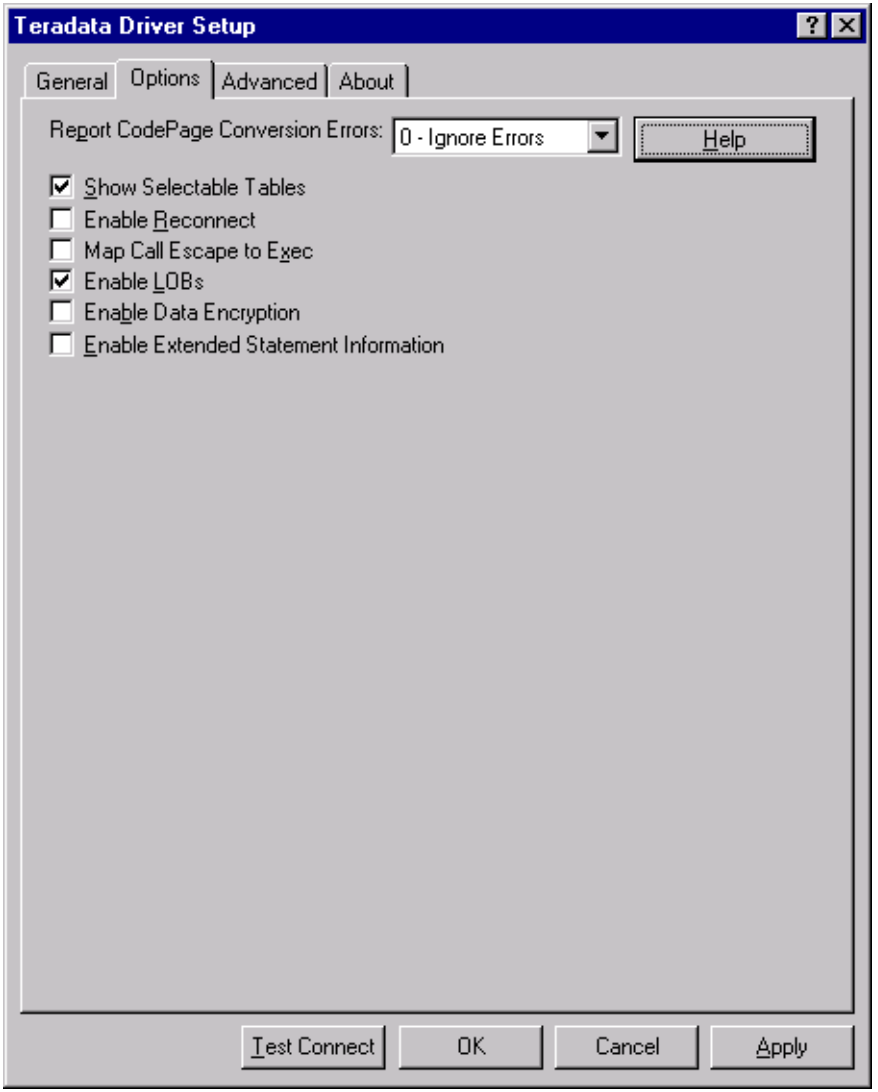


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 808)	None
Description (see page 811)	None
DBCName or Alias (see page 809)	None
DBCName List (see page 809)	None
Integrated Security (see page 815)	Disabled
Security Mechanism (see page 820)	None
Security Parameter (see page 823)	None
UserID (see page 825)	None
Default Database (see page 811)	None
Account String (see page 807)	None
Session Character Set (see page 823)	ASCII

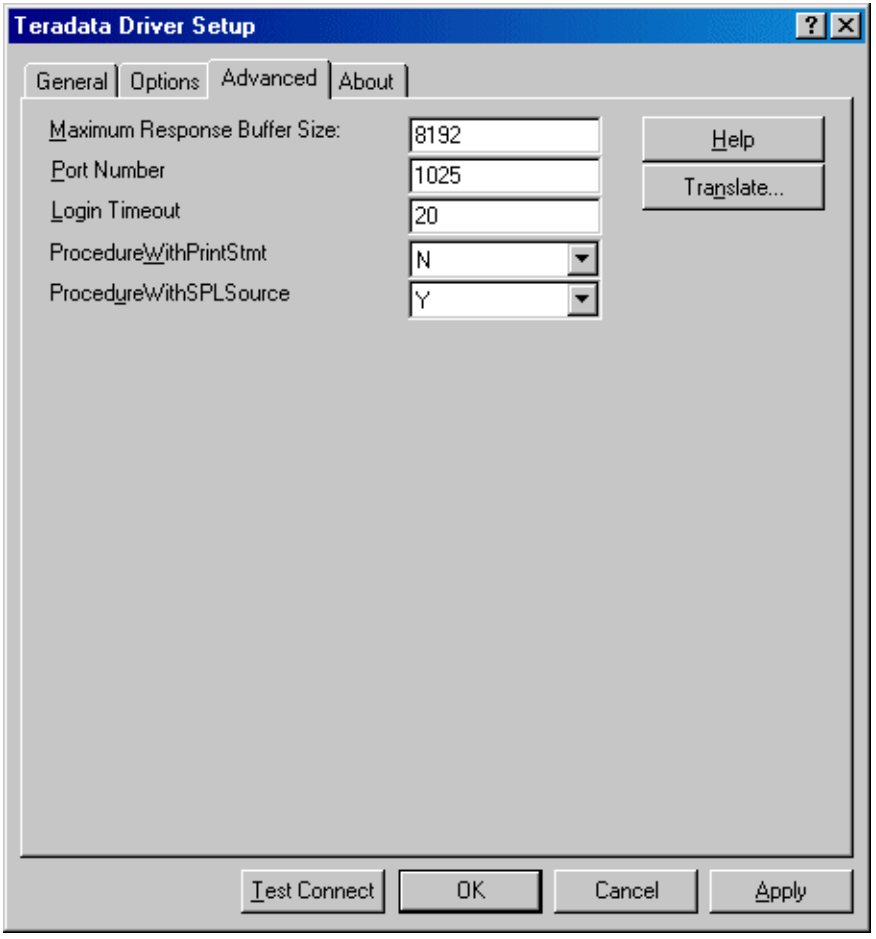
4 Click the **Options** tab to specify additional configuration options.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Options	Default
Report Codepage Conversion Errors (see page 819)	0 - Ignore Errors
Show Selectable Tables (see page 824)	Enabled
Enable Reconnect (see page 814)	Disabled
Map Call Escape to Exec (see page 816)	Disabled
Enable LOBs (see page 813)	Enabled
Enable Data Encryption (see page 812)	Disabled
Enable Extended Statement Information (see page 812)	Disabled

5 Optionally, click the **Advanced** tab to specify additional data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Maximum Response Buffer Size (see page 817)	8192
Port Number (see page 817)	1025
Login Timeout (see page 816)	20

Connection Options: Advanced *(cont.)*	Default
ProcedureWithPrintStmt (see page 818)	N (No)
ProcedureWithSPLSource (see page 818)	Y (Yes)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 803 for details). The information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a `Connection Established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message.

  Verify that all required client software is properly installed. If it is not, you will see the message:

  ```
  Specified driver could not be loaded due to system
  error [xxx].
  ```

  Click **OK**.

7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 15-1 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Teradata is:

```
DSN=Teradata Tables;AS=User2;EnableDataEncryption=Yes
```

A FILEDSN connection string is similar except for the initial keyword:

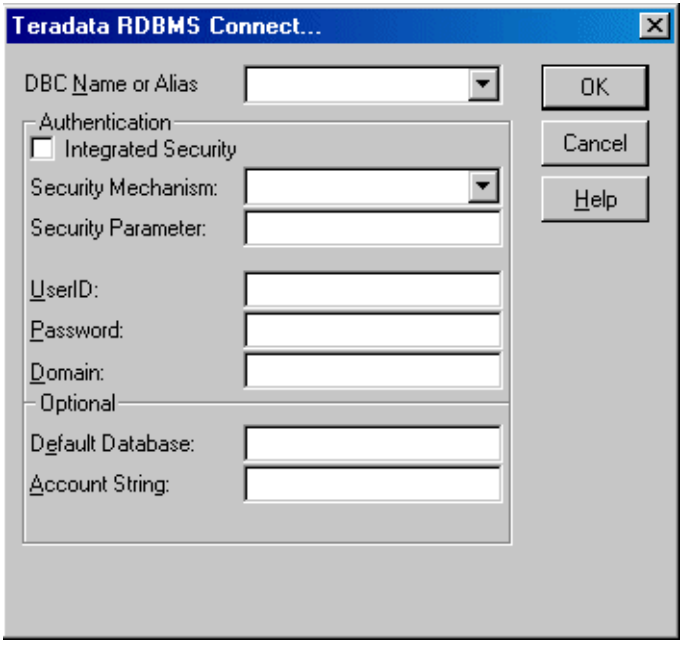
```
FILEDSN=Teradata.dsn;AS=User2;EnableDataEncryption=Yes
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect Teradata;DBCSN=123.456.78.90;UIS=YES
```

Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.



In this dialog box, provide the following information:

1. Select an alias name or IP address of a Teradata server from the DBC Name or Alias drop-down list. The choices for this list are determined by the entries in DBC Name or Alias and DBCName List on the General tab of the driver Setup dialog box.
2. Select the Integrated Security check box to enable the user to connect to the database through Single Sign On (SSO) using one of the authentication mechanisms that support SSO. In this case, User Name, Password, and Domain are not required and are not available fields.
3. If you do not use Integrated Security, select a value from the Security Mechanism drop-down list to specify the authentication mechanism used for connections to the data source.
4. Type a string of characters in the Security Parameter field that is to be regarded as a parameter to the authentication mechanism. The string is ignored by the ODBC driver and is passed on to the TeraSSO function that is called to set the authentication mechanism.

   The characters [] {} () , ; ? * = ! @ must be enclosed in curly braces.
5. Other options that are displayed on the Logon Dialog box depend on the authentication mechanism selected. See the descriptions of these options under Security Mechanism.
6. Type the domain name for Third Party Sign On along with the username and password. If a domain name is not provided, then the local domain is assumed.
7. Type a default Teradata database (optional).

8 Type an account string to be used during the creation of a user account in the Teradata Database instead of providing account information during configuration of ODBC (optional).

9 Click **OK** to complete the logon and to update these values in the Registry.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 15-1 lists the connection string attributes supported by the driver.

Table 15-1. Teradata Attribute Names

Attribute (Short Name)	Default
AccountString (AS)	None
AuthenticationDomain (AD)	None
AuthenticationPassword (AP)	None
AuthenticationUserId (AUI)	None
CharacterSet (CS)	ASCII
Database (DB)	None
DataSourceName (DSN)	None
DBCName (DBCN)	None
DBCName List	None
Description (n/a)	None
EnableDataEncryption (EDE)	No (Disabled)
EnableExtendedStmtInfo (EESI)	No (Disabled)
EnableLOBs (EL)	Yes (Enabled)
EnableReconnect (ER)	No (Disabled)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntegratedSecurity (IS)	No (Disabled)
LoginTimeout (LTO)	20
MapCallEscapeToExec (MCETE)	No (Disabled)
MaxRespSize (MRS)	8192
Password (PWD)	None
PortNumber (PORT)	1025
PrintOption (PO)	N (No)
ProcedureWithSPLSource (PWSS)	Y (Yes)
ReportCodepageConversionErrors (RCCE)	0 (Ignore Errors)
Security Mechanism (SECM)	None
SecurityParameter (SP)	None
ShowSelectableTables (SST)	Yes (Enabled)
TDProfile (TDP)	None
TDRole (TDR)	None

Table 15-1. Teradata Attribute Names (cont.)

Attribute (Short Name)	Default
TDUserName (TDUN)	None
UserID (UID)	None

Account String

Attribute: AccountString (AS)

Description: An account string. For a complete description of account strings, refer to the *Teradata Database Administration Guide*.

Valid Values: *string*

where *string* is an account string.

Default: None

GUI Tab: General tab on page 797

Authentication Password

Attribute: AuthenticationPassword (AP)

Description: The password for the Kerberos, LDAP, NTLM, and TD authentication mechanisms. The Authentication Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values: *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Authentication UserID

Attribute AuthenticationUserId (AUI)

Description The user ID for the Kerberos, LDAP, NTLM, and TD authentication mechanisms.

Valid Values *userid*

where *userid* is a valid user ID.

Default None

GUI Tab General tab on page 797

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 797

DBCName List

Attribute n/a

Description A list of IP addresses or aliases to appear in the drop-down list of the Logon dialog box (see "Using a Logon Dialog Box" on page 803 for a description). The DBCName List option is not used as a runtime connection attribute.

Valid Values *ip_address* | *alias* [,*ip_address* | *alias*][...]

where

ip_address is an IP address to appear in the drop-down list of the Logon dialog box.

alias is an alias to appear in the drop-down list of the Logon dialog box.

Separate multiple IP addresses or aliases with commas. The same restrictions apply as described for the DBCName or Alias option.

Default None

GUI Tab General tab on page 797

DBCName or Alias

Attribute DBCName (DBCN)

Description The IP address or alias of the Teradata server.

Valid Values *IP_address* | *alias*

where

IP_address is the IP address of the Teradata server.

alias is the alias of the Teradata server.

If set to *IP_address*, the time the driver waits for connections to be established is faster. The disadvantage is that if the server designated by that IP address is unavailable, the connection fails and the driver does not attempt to fail over to another IP address.

If set to *alias*, the time the driver waits for connections to be established is slower because the driver must search a local hosts file to resolve the alias to an IP address. The advantage is that the driver fails over the connection to an alternate IP address if the first address fails.

To use aliases, a local hosts file that maps aliases to IP addresses is required. Aliases cannot be more than eight characters. In the hosts file, you must specify the aliases and map each of them to an IP address in the order that you want the driver to attempt the connections. For example:

```
167.56.78.1 (NCR5100COP1)
167.56.78.2 (NCR5100COP2)
167.56.78.3 (NCR5100COP3)
```

where NCR5100 is an alias and COP*n* (where *n* = 1, 2, 3, ..., 128) is a suffix that sets the order of failover connection attempts. The eight-character limit on the alias does not include the suffix. You can enter a maximum of 128 COP (communications processor) entries per host.

NOTE: Although you must add a COP suffix to the alias in the hosts file, do *not* specify the suffix when entering the alias in the DBCName or Alias field of the Setup dialog box. Only specify the alias.

Default None

GUI Tab General tab on page 797

Default Database

Attribute Database (DB)

Description The name of the database to which you want to connect.

Valid Values *database_name*

where *database_name* is the name of a valid database.

Default None

GUI Tab General tab on page 797

Default Role

Attribute TDRole (TDR)

Description Specifies the Teradata role for the LDAP authentication mechanism.

Valid Values *string*

where *string* is a valid role.

Default None

GUI Tab General tab on page 797

Description

Attribute Description (n/a)

Description 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 on page 797

Enable Data Encryption

Attribute EnableDataEncryption (EDE)

Description Determines whether the driver uses data encryption.

Valid Values Yes | No

If set to Yes (Enabled), the driver encrypts data and communicates with the Teradata gateway using encryption.

NOTE: Before you use this value, verify that the server is encryption capable. Data encryption may adversely affect performance because of the additional overhead (mainly CPU usage) required to encrypt and decrypt data.

If set to No (Disabled), the driver does not encrypt data except for logon information.

Default No (Disabled)

GUI Tab Options tab on page 798

Enable Extended Statement Information

Attribute EnableExtendedStmtInfo (EESI)

Description Determines whether the driver supports extended statement information.

Valid Values Yes | No

If set to Yes (Enabled), the driver queries the server to see if it supports the Statement Information parcel. If the server supports the Statement Information parcel, the driver requests the Statement Information parcel and enables auto-generated key retrieval and SQLDescribeParam support. Use this value if you want to enable the Return Generated Keys option.

If set to No (Disabled), the driver does not attempt to expose auto-generated key retrieval or SQLDescribeParam.

Default No (Disabled)

GUI Tab Options tab on page 798

Enable LOBs

Attribute EnableLOBs (EL)

Description Determines whether the driver enforces native LOB data type mapping.

Valid Values Yes | No

If set to Yes (Enabled), the driver enforces native LOB data type mapping as described:

- ODBC data type SQL_LONGVARBINARY is mapped to the Teradata BLOB feature.
- ODBC data type SQL_LONGVARCHAR is mapped to the Teradata CLOB feature.

If set to No (Disabled), the driver provides backward compatibility for applications without LOB support that are using a version of Teradata Database prior to V2R5.1. The mappings are:

- ODBC data type SQL_LONGVARBINARY is mapped to the Teradata VARBYTE(32000) feature.
- ODBC data type SQL_LONGVARCHAR is mapped to the Teradata LONG VARCHAR feature.

This value can improve performance if your application does not send data to, or retrieve it from, LOB columns. You may receive an error if you disable this option and try to retrieve data from a LOB column.

Default Yes (Enabled)

GUI Tab Options tab on page 798

Enable Reconnect

Attribute EnableReconnect (ER)

Description Determines whether the driver will reconnect after a system crash or reset is detected.

Valid Values Yes | No

If set to Yes (Enabled), the driver attempts to reconnect to the saved sessions; however, sessions cannot be reconnected until the Teradata system is available. After a session has been reconnected, applications can expect to receive error messages describing why the ODBC function failed, as well as a status report describing the post-recovery state.

If set to No (Disabled), the driver does not attempt to reconnect to the saved sessions.

Default No (Disabled)

GUI Tab Options tab on page 798

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description 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. Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details.

The driver uses the specified IANA code page to convert "W" (wide) functions to ANSI.

The driver and Driver Manager both check for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

If the driver does not find an IANAAppCodePage value, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 800

Integrated Security

Attribute IntegratedSecurity (IS)

Description Determines whether the driver allows the user to connect to the database using Single Sign On (SSO) through an authentication mechanism that supports SSO.

Valid Values Yes | No

If set to Yes (Enabled), SSO is allowed. The driver uses the operating system user ID and password.

If set to No (Disabled), you must specify a value for the UserID option.

Default No (Disabled)

GUI Tab General tab on page 797

Login Timeout

Attribute: LoginTimeout (LTO)

Description: The number of seconds to wait when establishing a virtual circuit with Teradata for login.

Valid Values: *x*

where *x* is a positive integer.

If set to *x*, the driver waits the specified number of seconds.

Default: 20

GUI Tab: Advanced tab on page 800

Map Call Escape to Exec

Attribute: MapCallEscapeToExec (MCETE)

Description: Determines whether the driver converts the `{CALL <name>(...)}` statement to `EXEC name(...)`.

Valid Values: Yes | No

If set to Yes (Enabled), the driver considers the `{CALL <name>(...)}` statement as the SQL for MACRO execution and converts it to `EXEC name(...)`.

If set to No (Disabled), the driver does not convert `{CALL name(...)}` statements to `EXEC name(...)`, and considers them as CALL statements for Stored Procedure Execution.

Default: No (Disabled)

GUI Tab: Options tab on page 798

Maximum Response Buffer Size

Attribute MaxRespSize (MRS)

Description The size of the Teradata response buffer used for SQL requests. This value may be adjusted dynamically if Teradata cannot send a result within the defined size.

Valid Values A positive integer from 1 to 65477

If using a slow TCP/IP interface, such as PPP or SLIP, enter a smaller value. If you expect to retrieve large result sets in a LAN environment, set a larger value.

Default 8192

GUI Tab Advanced tab on page 800

Password

Attribute Password (PWD)

Description The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values *pwd*

where *pwd* is a valid password.

Default None

GUI Tab n/a

Port Number

Attribute PortNumber (PORT)

Description The port number of the server listener.

Valid Values *port_name*

where the *port_name* is the port number of the server listener. Check with your database administrator for the correct number.

Default 1025

GUI Tab Advanced tab on page 800

ProcedureWithPrintStmt

Attribute PrintOption (PO)

Description Determines whether the driver activates the print option when creating stored procedures.

Valid Values P | N

If set to P (Print), the driver activates the print option.

If set to N (No), the driver does not activate the print option.

Default N (No)

GUI Tab Advanced tab on page 800

ProcedureWithSPLSource

Attribute ProcedureWithSPLSource (PWSS)

Description Determines whether the drive specifies SPL text when creating stored procedures.

Valid Values Y | N

If set to Y (Yes), the driver specifies SPL text.

If set to N (No), the driver does not specify SPL text.

Default Y (Yes)

GUI Tab Advanced tab on page 800

Profile

Attribute TDProfile (TDP)

Description Specifies the Teradata profile for the LDAP authentication mechanism.

Valid Values *string*

where *string* is a valid profile.

Default None

GUI Tab General tab on page 797

Realm

Attribute AuthenticationDomain (AD)

Description Specifies the domain appropriate to the selected authentication mechanism

Valid Values *string*

where *string* is a valid domain.

Default None

GUI Tab None

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors (RCCE)

Description 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

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 on page 800

Security Mechanism

Attribute Security Mechanism (SECM)

Description The authentication method to be used by the driver for connections to the database.

Valid Values TD1 | TD2 | LDAP | KRB5 | KRB5C | NTLM | NTLMC

If set to TD1, the driver uses Teradata 1.

If set to TD2, the driver uses Teradata 2.

If set to LDAP, the driver uses LDAP.

If set to KRB5, the driver uses Kerberos on Windows clients working with Windows servers if the server is V2R6.0.

If set to KRB5C, the driver uses Kerberos Compatibility on Windows clients working with Windows servers if the server is pre-V2R6.0.

If set to NTLM, the driver uses NTLM on Windows clients working with Windows servers if the server is V2R6.0.

If set to NTLMC, the driver uses NTLM Compatibility on Windows clients working with Windows servers if the server is pre-V2R6.0.

NOTE: Kerberos and NTLM are enabled through the Teradata client. See your Teradata documentation for requirements.

The following options may display, based on the selected method:

- *No mechanism selected*
  - **User name**: A user name for the default Teradata database. If TeraSSO allows fully qualified user names, the user name may contain a domain or realm, for example, {judy@linedata}. Values containing a character like @ must be enclosed in braces.
- *KRB5 and KRB5C*
  - **Authentication UserID**: The Kerberos user ID.
  - **Realm**: The Kerberos domain. (The equivalent connection string attribute is AuthenticationDomain.)
- *LDAP*
  - **Authentication UserID**: The LDAP user ID.
  - **Realm**: The LDAP domain. (The equivalent connection string attribute is AuthenticationDomain.)
  - **TD User name**: The Teradata user name.
  - **Profile**: The Teradata Profile. (The equivalent connection string attribute is TDProfile.)

- **Default Role**: The Teradata Role. (The equivalent connection string attribute is TDRole.)

- *NTLM and NTLMC*
  - **Authentication UserID**: The NTLM user ID.
  - **Realm**: The NTLM domain. (The equivalent connection string attribute is AuthenticationDomain.)
- *TD1 and TD2*
  - **Authentication UserID**: The TD1 or TD2 user ID.

Other parameters for the authentication mechanism can be entered in the Security Parameter field.

Default None

GUI Tab General tab on page 797

Security Parameter

Attribute SecurityParameter (SP)

Description A string that is passed as a parameter to the authentication method. The string is ignored by the ODBC driver and is passed to the TeraSSO function that is called to set the authentication method.

Valid Values *string*

where *string* is a string of characters. The characters [] {} () , ; ? * = ! @ must be enclosed in curly braces.

Default None

GUI Tab General tab on page 797

Session Character Set

Attribute CharacterSet (CS)

Description A character set used to override the Teradata character set.

Valid Values ASCII | UTF16 (valid only for V2R6.*x* servers) | LATIN1252_0A | LATIN9_0A | LATIN1_0A | Shift-JIS | EUC | BIG5 | GB | NetworkKorean

The specified character set must be installed on the database.

Default ASCII

GUI Tab General tab on page 797

Show Selectable Tables

Attribute ShowSelectableTables (SST)

Description Determines whether the driver supports X views.

Valid Values Yes | No

If set to Yes (Enabled), SQLTables() and SQLProcedures() use dbc.tablesX and dbc.databasesX instead of dbc.tables and dbc.databases. Also, SQLColumns() and SQLProcedureColumns() use dbc.columnsX instead of dbc.columns. SqlStatistics() uses dbc.tablesizeX instead of dbc.tablesize. The X tables only contain information that the user has permission to access. These tables are optional for Teradata, so verify that they exist.

If set to No (Disabled), SQLTables() and SQLProcedures() use dbc.tables and dbc.databases. Also, SQLColumns() and SQLProcedureColumns() use dbc.columns. SqlStatistics() uses dbc.tablesize.

Default Yes (Enabled)

GUI Tab Options tab on page 798

TDUserName

Attribute TDUserName (TDUN)

Description Specifies the Teradata user name for the LDAP authentication mechanism.

Valid Values *user_name*

where *user_name* is a valid user name.

Default None

GUI Tab General tab on page 797

UserID

Attribute UserID (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.

The user name is interpreted in the context of the authentication mechanism. If, for example, the authentication mechanism is NTLM, the user name is assumed to be a Windows user name.

If TeraSSO allows fully qualified user names, the user name may contain a domain or realm, for example, {judy@linedata}. Values containing a character such as @ must be enclosed in curly braces.

SSO is indicated by the absence of a UserID.

Default None

GUI Tab General tab on page 797

Data Types

Table 15-2 shows how the Teradata data types map to the standard ODBC data types.

Table 15-2. Teradata Data Types

Teradata	ODBC
Blob [1]	SQL_LONGVARBINARY
Bigint [2]	SQL_BIGINT
Byte	SQL_BIT
Byteint	SQL_TINYINT
Char	SQL_CHAR
Clob [3]	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Decimal [4]	SQL_DECIMAL
Double	SQL_DOUBLE
Float	SQL_FLOAT
Integer	SQL_INTEGER
Interval day	SQL_INTERVAL_DAY
Interval day to hour	SQL_INTERVAL_DAY_TO_HOUR
Interval day to minute	SQL_INTERVAL_DAY_TO_MINUTE
Interval day to second	SQL_INTERVAL_DAY_TO_SECOND
Interval hour	SQL_INTERVAL_HOUR
Interval hour to minute	SQL_INTERVAL_HOUR_TO_MINUTE
Interval hour to second	SQL_INTERVAL_HOUR_TO_SECOND

***Table 15-2. Teradata Data Types** (cont.)*

Teradata	ODBC
Interval minute[5]	SQL_INTERVAL_MINUTE
Interval minute to second	SQL_INTERVAL_MINUTE_TO_SECOND
Interval month[5]	SQL_INTERVAL_MONTH
Interval second	SQL_INTERVAL_SECOND
Interval year	SQL_INTERVAL_YEAR
Interval year to month	SQL_INTERVAL_YEAR_TO_MONTH
Numeric	SQL_NUMERIC
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR

1. If no LOB support, VARBYTE(32000).
2. Supported only on Teradata 6.2 and higher.
3. If no LOB support, LONGVARCHAR.
4. Precision of 18 unless on a Teradata 6.2 or higher server that supports large decimal types.
5. Supported only on Teradata 6.2 and higher when EnableExtendedStmtInfo is enabled.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Unicode Support

The driver supports Unicode data types. Table 15-3 shows how the Teradata data types map to the Unicode data types, but only when CharacterSet is set to UTF-16.

Table 15-3. Teradata Unicode Data Types

Teradata	Unicode
char () charset Unicode	SQL_WCHAR
clob charset Unicode	SQL_WLONGVARCHAR
varchar () charset Unicode	SQL_WVARCHAR

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

Teradata supports isolation levels 0 (read uncommitted) and 3 (serializable).

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the driver for the Teradata database.

Number of Connections and Statements Supported

The driver supports multiple connections and 16 statements per connection to the Teradata database system.

Part 3: The 32-Bit Drivers

This part describes the drivers that are available only in 32-bit versions. See "Part 2: The 32-Bit/64-Bit Drivers" on page 171 for the drivers that are available in both 32- and 64-bit versions.

This part contains the following chapters:

- Chapter 16 "The Btrieve (Pervasive.SQL) Driver" on page 833
- Chapter 17 "The dBASE Driver" on page 861
- Chapter 18 "The Informix Driver" on page 899
- Chapter 19 "The Paradox Driver" on page 929
- Chapter 20 "The Text Driver" on page 957
- Chapter 21 "The XML Driver" on page 991

16 The Btrieve (Pervasive.SQL) Driver

The DataDirect Connect *for* ODBC Btrieve driver (the Btrieve driver) supports the following versions of Btrieve files:

- Pervasive.SQL 2000
- Pervasive.SQL 8.5
- Pervasive.SQL 7.0
- Btrieve version 6.15

The driver executes SQL statements directly on Btrieve files.

The Btrieve driver is 32-bit only and is supported in the Windows environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Btrieve driver.

Driver Requirements

To access a Btrieve database, you must be using the appropriate client software for the version of the Btrieve database to which you are connecting:

Database Versions	Client Names
Pervasive.SQL 2000	Pervasive.SQL 2000 client software
Pervasive.SQL 8.5	Pervasive.SQL 8.5 client software
Pervasive.SQL 7.0	Pervasive.SQL 7.0 client software
Btrieve 6.15 for Windows 9*x*	Btrieve Developer's Kit or Btrieve WorkStation Client Engine
Btrieve 6.15 for Windows NT	Btrieve Developer's Kit, Btrieve WorkStation Client Engine, or Btrieve Client/Server Database Engine

NOTE: The Btrieve driver may experience problems if the Btrieve Microkernel Engine's communication buffer size is smaller than that of the Btrieve driver's Array Size option. You can increase the communication buffer size with the Pervasive Software Setup Utility, or you can decrease the value of Array Size option through the ODBC Btrieve Driver setup dialog box or through the ArraySize connection string attribute.

Before you attempt to access Btrieve files, you must incorporate existing Btrieve files into a Scalable SQL database. See "Managing Databases" on page 835 for information about Scalable SQL databases.

Managing Databases

If you already use Scalable SQL, the Btrieve driver can access your Scalable SQL databases directly. If not, your Btrieve files must be incorporated into a Scalable SQL database.

A Scalable SQL database is composed of data files that contain your records and data dictionary files that describe the database. The data files are Btrieve files. The data dictionary files are special Btrieve files that contain descriptions of the data files, views, fields, and indexes in your database.

All Btrieve files in a Scalable SQL database must reside in the same directory. In addition to the Btrieve data files, the three data dictionary files (FILE.DDF, FIELD.DDF, and INDEX.DDF) also must be in the directory.

Incorporating a Btrieve file into a Scalable SQL database does not change the Btrieve file in any way. You can continue to access the file directly with any existing Btrieve application.

Transactions

The Btrieve driver supports *transactions*. A transaction is a series of database changes that is treated as a single unit. In applications that so not use transactions, the Btrieve driver immediately executes Insert, Update, and Delete statements on the database files and the changes are automatically committed when the SQL statement is executed. You cannot undo these changes. In applications that use transactions, the Btrieve driver holds inserts, updates, and deletes until you issue a Commit or Rollback. A Commit saves the changes to the database file; a Rollback undoes the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

To use the Btrieve driver's transaction processing capabilities, consult the Pervasive documentation.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 841 and Table 16-1 on page 843 for an alphabetical list of driver connection string attributes and their initial default values.

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override

the default values of the data source if you want to change these values at connection time.

To configure a Btrieve data source:

1 Start the ODBC Administrator by selecting its icon from the DataDirect Connect program group; then, select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name on the User DSN tab and click **Configure** to display the driver Setup dialog box.

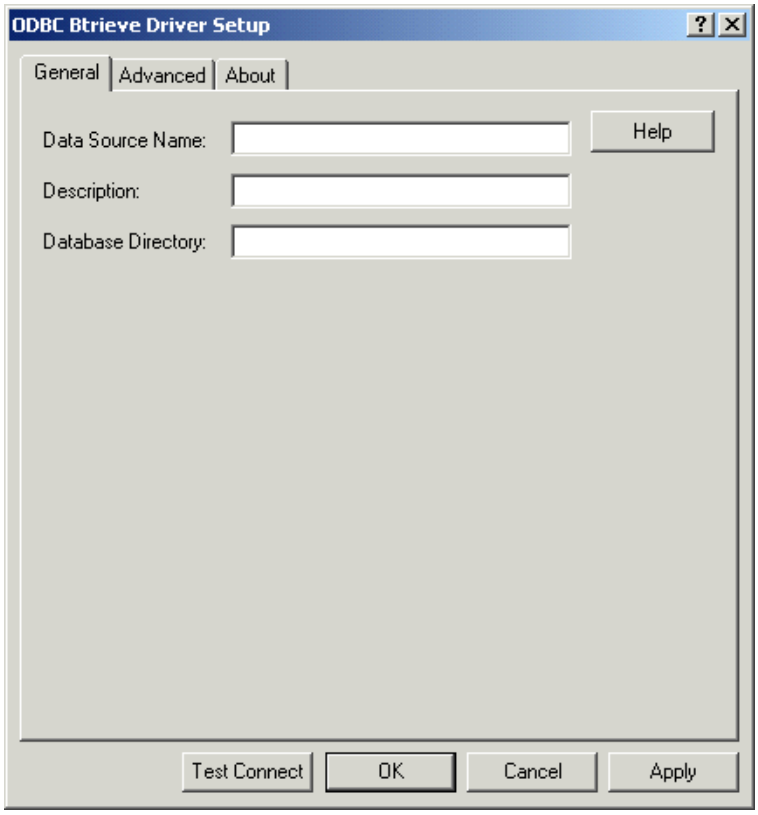
  If you are configuring a new user data source, click **Add** on the User DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **System DSN**: To configure a new system data source, click **Add** on the System DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source name on the File DSN tab and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** on the File DSN tab to display a list of installed drivers. Select the driver and click **Next**. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

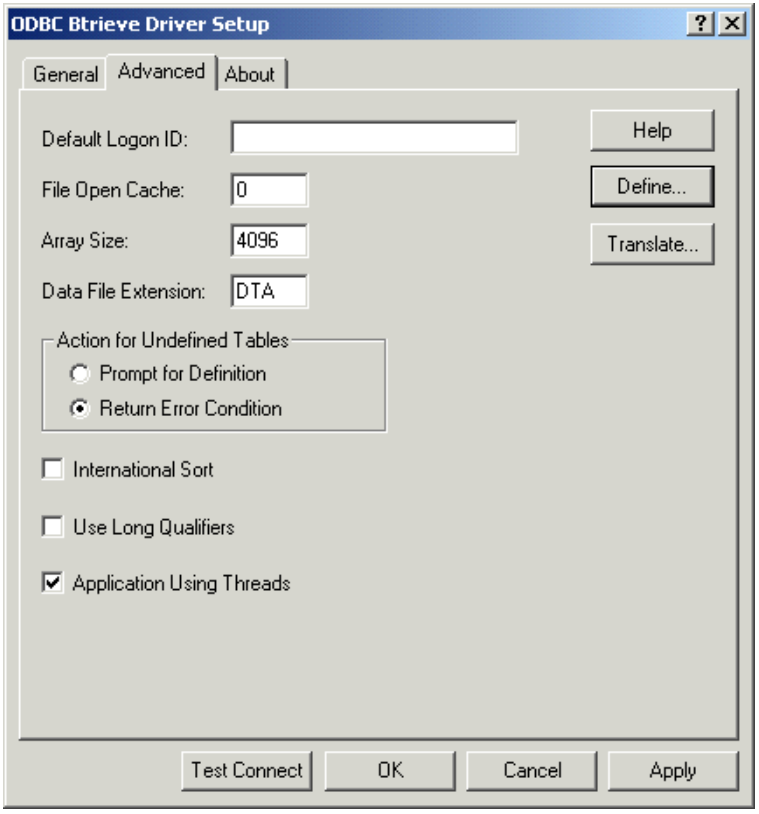


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

2 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 845)	None
Description (see page 847)	None
Database Directory (see page 846)	None

3 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Default Logon ID (see page 846)	None
File Open Cache (see page 847)	0
Array Size (see page 844)	4096
Data File Extension (see page 845)	DTA
Action for Undefined Tables (see page 843)	Return Error Condition
International Sort (see page 848)	Disabled

Connection Options: Advanced	Default
Use Long Qualifiers (see page 849)	Disabled
Application Using Threads (see page 844)	Enabled

Define: Click **Define** to define table structure as described in "Defining Table Structure" on page 849.

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

4 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

- If the driver can connect, it releases the connection and displays a `Connection established!` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message.

  Verify that all required client software is properly installed. If it is not, you will see the message:

  ```
  Specified driver could not be loaded due to system
  error [xxx].
  ```

  Click **OK**.

5 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 16-1 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Btrieve is:

```
DSN=BTRIEVE FILES;DB=J:\Btrvdata
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Btrieve.dsn;DB=J:\Btrvdata
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 Btrieve;
DB=J:\Btrvdata;UID=JOHN;PWD=XYZZY
```

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 16-1 lists the connection string attributes supported by the Btrieve driver.

Table 16-1. Btrieve Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
ArraySize (AS)	4096
Database (DB)	None
DataFileExtension (DFE)	DTA
DataSourceName (DSN)	None
Description (n/a)	None
FileOpenCache (FOC)	0 (No File Open Caching)
IntlSort (IS)	0 (Disabled)
LogonID (UID)	None
Password (PWD)	None
UndefinedTable (UT)	Error
UseLongQualifiers (ULQ)	0 (Disabled)

Action for Undefined Tables

Attribute: UndefinedTable (UT)

Description: Determines whether the driver prompts the user when it encounters a table for which it has no structure information.

Valid Values: PROMPT | ERROR

Specify PROMPT to prompt the user.

Specify ERROR to return an error.

Default ERROR (driver returns an error)

GUI Tab Advanced tab on page 839

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI Tab Advanced tab on page 839

Array Size

Attribute ArraySize (AS)

Description The number of bytes in the array. This connection option enables the driver to retrieve an array of records from the Btrieve database and, in most cases, results in improved performance for the application.

Valid Values A positive integer from 1 to 65535

Default 4096

GUI Tab Advanced tab on page 839

Data File Extension

Attribute DataFileExtension (DFE)

Description A one- to three-character file name extension to use for data files.

Valid Values *ext*

where *ext* is the name of the one- to three-character file name extension.

This value is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option causes an error.

In other SQL statements, such as Select or Insert, users can specify an extension other than the one specified for this connection option. The Data File Extension value is used when no extension is specified.

Default DTA

GUI Tab Advanced tab on page 839

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 838

Database Directory

Attribute: Database (DB)

Description: The directory that contains the data files.

Valid Values: *database_directory*

where *database_directory* is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

This includes both Btrieve files and the data dictionary files (.DDF). Data dictionary files describe the structure of Btrieve data.

Default: None

GUI Tab: General tab on page 838

Default Logon ID

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: Advanced tab on page 839

Description

Attribute: Description (n/a)

Description: 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 on page 838

File Open Cache

Attribute: FileOpenCache (FOC)

Description: The maximum number of used file handles to cache.

Valid Values: 0 | *x*

where *x* is a positive integer.

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default: 0 (No File Open Caching)

GUI Tab: Advanced tab on page 839

International Sort

Attribute: IntlSort (IS)

Description: Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values: 0 | 1

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default: 0 (Disabled)

GUI Tab: Advanced tab on page 839

Password

Attribute: Password (PWD)

Description: The password that you must enter if your Scalable SQL data dictionary files have security restrictions set. The Password option cannot be specified through the Administrator GUI.

Valid Values: *pwd*

where *pwd* is a valid password.

Default: None

GUI Tab: n/a

Use Long Qualifiers

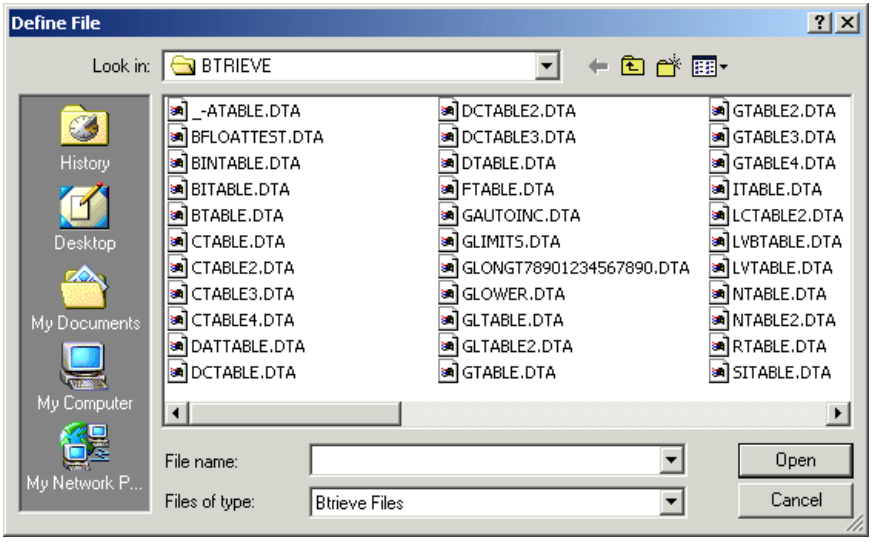
Attribute	UseLongQualifiers (ULQ)
Description	Determines whether the driver uses long path names.
Valid Values	0 | 1 If set to 1 (Enabled), path names can be a maximum of 255 characters. If set to 0 (Disabled), path names can be a maximum of 128 characters.
Default	0 (Disabled)
GUI Tab	Advanced tab on page 839

Defining Table Structure

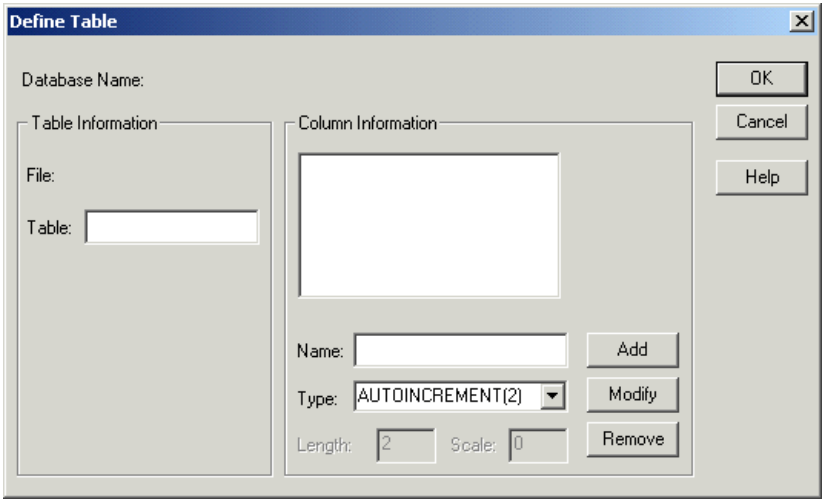
Because Btrieve does not store any column information in the data file, you may need to define its structure. Tables created by the Btrieve driver or by Scalable SQL will not require this. Utilities are also available from Pervasive Software that will perform this operation.

To define the structure of a file:

1 Display the ODBC Btrieve Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.



2 In the Define File dialog box, select the file you want to define and click **Open** to display the Define Table dialog box.



Database Name: This field displays the directory in which the file you selected in the Define File dialog box is located.

File: This field displays the name of the file that you selected in the Define File dialog box.

Table: Type the name of the table to be returned by SQLTables. The name can be a maximum of 20 characters and cannot be the same as another defined table in the database. This field is required.

3 Specify values in the following fields to define each column. Click **Add** to add the column name to the list box.

Name: Type the name of the column.

Type: Select the data type of the column.

Length: Type the length of the column, if applicable.

Scale: Type the scale of the column, if applicable.

4 To modify an existing column definition, select the column name in the list box. Modify the values for that column name; then, click **Modify**.

5 To delete an existing column definition, select a column name in the list box and click **Remove**.

6 Click **OK** to define the table.

Data Types

Table 16-2 shows how the Btrieve data types map to the standard ODBC data types. The Btrieve data types are used when you incorporate Btrieve files into a Scalable SQL database.

Table 16-2. Btrieve Data Types

Btrieve	ODBC
Autoincrement(2)	SQL_SMALLINT
Autoincrement(4)	SQL_INTEGER
Bfloat(4)	SQL_REAL
Bfloat(8)	SQL_DOUBLE
Bit	SQL_BIT
Blob	SQL_LONGVARGBINARY
Char	SQL_CHAR
Currency	SQL_DECIMAL
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Float(4)	SQL_REAL
Float(8)	SQL_DOUBLE
Integer(1)	SQL_TINYINT
Integer(2)	SQL_SMALLINT
Integer(4)	SQL_INTEGER

Table 16-2. Btrieve Data Types (cont.)

Btrieve	ODBC
Integer(8)	SQL_BIGINT
Logical(1)	SQL_BIT
Logical(2)	SQL_BIT
Lstring	SQL_VARCHAR
Money	SQL_DECIMAL
Note	SQL_LONGVARCHAR
Numeric	SQL_NUMERIC
Numericsts	SQL_NUMERIC
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Unsigned(1)	SQL_TINYINT
Unsigned(8)	SQL_BIGINT
Zstring	SQL_VARCHAR

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Indexes

NOTE: If you define an index using the Btrieve driver, the index will not have the restrictions discussed here.

For query optimization, the Btrieve driver does not use the following:

- Indexes containing all-segment-null keys or any-segment-null keys.
- Any index key that is marked case-insensitive.
- Any index keys where the data type of the index key does not match the data type of the field. The one exception is if the index key is declared as an unsigned integer and the field in the file is declared as signed integer, or vice versa, then the driver assumes the field contains only unsigned quantities and uses the index. Note that this can lead to incorrect results if the field in fact does contain signed quantities.

The Btrieve driver only uses an alternate-collating-sequence (ASC) index key for equality lookups. Additionally, if an ASC key is part of a segmented index, the other index segments are not used for query optimization unless the Where clause contains an equality condition for the ASC key.

Column Names

Column names in SQL statements (such as Select and Insert) can be up to 20 characters long. If column names are in all lowercase, a combination of upper and lowercase, contain blank spaces, or are reserved words, they must be surrounded by the grave character (`) (ASCII 96). For example:

```
SELECT `name` FROM emp
```

Select Statement

You use the SQL Select statement to specify the columns and records to be read. Btrieve Select statements support all the Select statement clauses described in Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference*. This section describes the information that is specific to Btrieve.

Rowid Pseudo-Column

Each Btrieve record contains a special column named Rowid. This field contains a unique number that indicates the record's sequence in the database. You can use Rowid in Where and Select clauses.

Rowid is particularly useful when you are updating records. You can retrieve the Rowid of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM emp
```

Then, you can use the Rowid of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=21
```

The fastest way of updating a single row is to use a Where clause with the Rowid. You cannot update the Rowid column.

Select statements that use the Rowid pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21    //fast search

SELECT * FROM emp WHERE rowid <=25  //full table scan
```

Alter Table Statement

The Btrieve driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_name data_type
| ADD (column_name data_type [, column_name data_type]...)
| DROP [COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept char 10)
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

Create and Drop Index Statements

The Btrieve driver supports SQL statements to create and delete indexes.

Create Index

The Create Index statement for Btrieve files has the form:

```
CREATE [UNIQUE] INDEX index_name ON table_name ([field_name
[ASC | DESC] [,field_name
[ASC | DESC]]...)
```

Unique means that Btrieve does not let you insert two records with the same index values.

index_name is the name of the index.

table_name is the name of the table on which the index is to be created.

ASC tells Btrieve to create the index in ascending order. DESC tells Btrieve to create the index in descending order. By default, indexes are created in ascending order. For example:

```
CREATE INDEX lname ON emp (last_name)
```

Drop Index

The form of the Drop Index statement is:

```
DROP INDEX table_name.index_name
```

table_name is the name of the table from which the index is to be dropped.

index_name is the name of the index.

For example:

```
DROP INDEX emp.lname
```

Isolation and Lock Levels Supported

Btrieve supports isolation level 1 (read committed) only. Btrieve supports record-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The Btrieve driver supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver supports the minimum SQL grammar with several core extensions.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Btrieve driver. In addition, the following function is supported: SQLSetPos.

Number of Connections and Statements Supported

Btrieve files support a single connection and multiple statements per connection.

17 The dBASE Driver

The DataDirect Connect *for* ODBC dBASE driver (the dBASE driver) supports the following file types:

- dBASE IV and V
- Clipper
- FoxPro 6.0 with 3.0 functionality only
- FoxPro 2.5, 2.6, and 3.0
- FoxPro 3.0 database container (DBC)

The dBASE driver runs the SQL statements directly on dBASE- and FoxPro-compatible files. You do not need to own dBASE or FoxPro products to access these files. The dBASE driver cannot access files that are larger than 2 GB.

The dBASE driver is 32-bit only and is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the dBASE driver.

Driver Requirements

There are no client requirements for the dBASE driver.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 872 and Table 17-1 on page 874 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and

storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 17-1 on page 874 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

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.

dBASE

To configure a dBASE data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

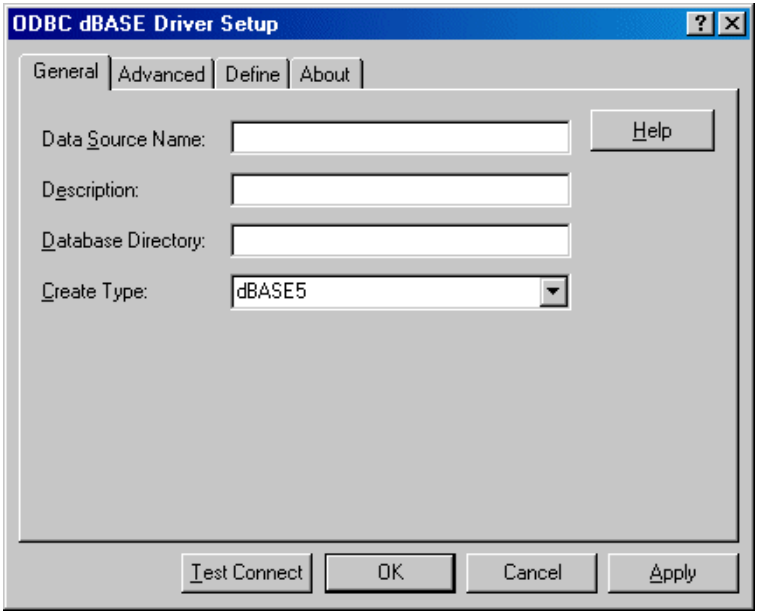
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

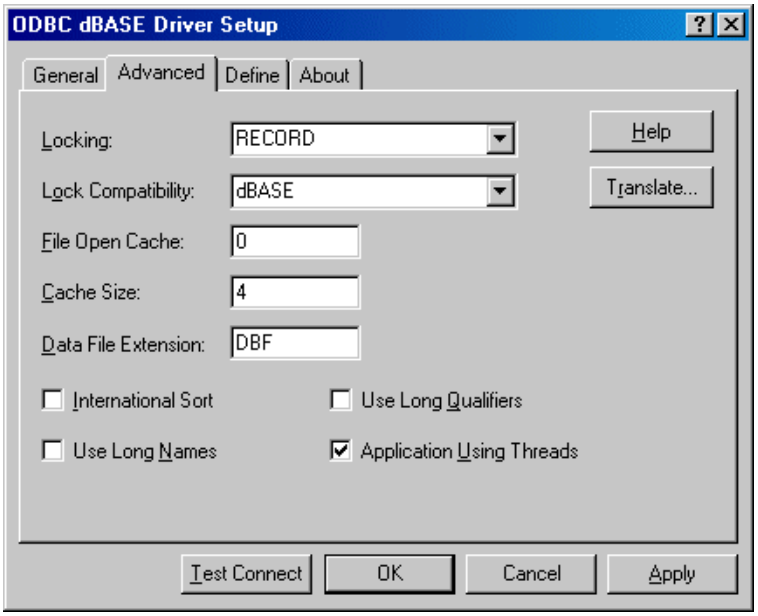


NOTE: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 877)	None
Description (see page 878)	None
Database Directory (see page 877)	None
Create Type [dBASE] (see page 876)	dBASE5

4 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Locking (see page 882)	RECORD
Lock Compatibility (see page 881)	dBASE
File Open Cache (see page 879)	0
Cache Size (see page 875)	4
Data File Extension (see page 876)	DBF
International Sort (see page 880)	Disabled
Use Long Names (see page 883)	Disabled
Use Long Qualifiers (see page 883)	Disabled
ApplicationUsingThreads (see page 874)	Enabled
IANAAppCodePage (see page 880) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

5 If you use index files that have different names than their corresponding data files and you have not defined this association, click the **Define** tab. See "Using a Connection String" on page 872 for step-by-step instructions.

6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

- If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

FoxPro 3.0 DBC

To configure a FoxPro 3.0 database container data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

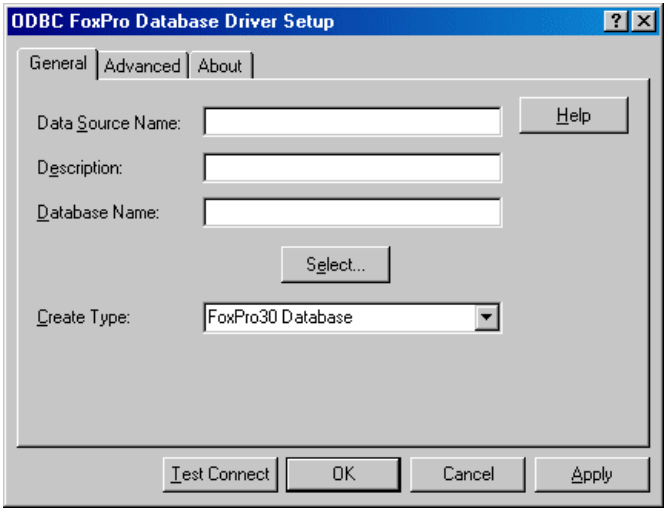
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.



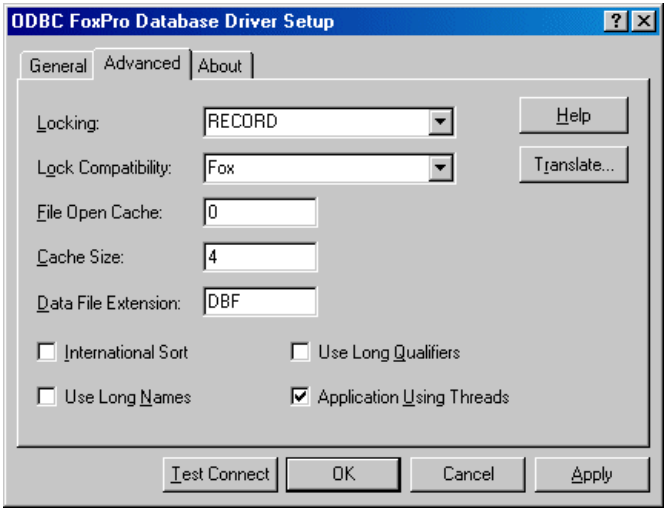
NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 877)	None
Description (see page 878)	None
Database Name (see page 878)	None
Create Type [FoxPro] (see page 876)	FoxPro30 Database

Click **Select** to choose the directory and .DBC file that you want to use.

4 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Locking (see page 882)	RECORD
Lock Compatibility (see page 881)	Fox
File Open Cache (see page 879)	0
Cache Size (see page 875)	4
Data File Extension (see page 876)	DBF

Connection Options: Advanced *(cont.)*	Default
International Sort (see page 880)	Disabled
Use Long Names (see page 883)	Disabled
Use Long Qualifiers (see page 883)	Disabled
ApplicationUsingThreads (see page 874)	Enabled
IANAAppCodePage (see page 880) UNIX ONLY	4 (ISO 8559-1 Latin-1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

5 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

- If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 17-1 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for dBASE is:

```
DSN=DBASE FILES;LCK=NONE;IS=0
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=DBASE.dsn;LCK=NONE;IS=0
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 dBASEFile (*.dbf);
DB=C:\DBASE;CT=dBASE5
```

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 17-1 lists the connection string attributes supported by the dBASE driver.

Table 17-1. dBASE Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
CacheSize (CSZ)	4
Create Type (CT) [dBASE]	dBASE5
Create Type (CT)) [FoxPro]	FoxPro30 Database
Database (DB) [dBASE]	None
Database (DB) [FoxPro]	None
DataFileExtension (DFE)	DBF
DataSourceName (DSN)	None
Description (n/a)	None
ExtensionCase (EC)	UPPER
FileOpenCache (FOC)	0 (no file open caching)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntlSort (IS)	0 (Disabled)
LockCompatibility (LCOMP)	dBASE
Locking (LCK)	RECORD
UseLongNames (ULN)	0 (Disabled)
UseLongQualifiers (ULQ)	0 (Disabled)

ApplicationUsingThreads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Cache Size

Attribute CacheSize (CSZ)

Description The number of 64 KB blocks the driver uses to cache database records. The larger the number of blocks, the better the performance.

Valid Values 0 | *x*

where *x* is a positive integer that specifies the number of 64 KB blocks for caching.

If set to 0, no records are cached.

If set to *x*, the specified number of 64 KB blocks are set aside for caching. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you are not able to see updates made by other users until you run the Select statement again.

Default 4

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Create Type [dBASE]

Attribute	Create Type (CT) [dBASE]
Description	The type of table or index to be created on a Create Table or Create Index statement.
Valid Values	dBASE4 | dBASE5 | Clipper | FoxPro25 | FoxPro30
Default	dBASE5
GUI tab	General tab on page 866

Create Type [FoxPro]

Attribute	Create Type (CT)) [FoxPro]
Description	The type of table or index to be created on a Create Table or Create Index statement.
Valid Value	FoxPro30 Database
Default	FoxPro30 Database
GUI tab	General tab on page 869

Data File Extension

Attribute	DataFileExtension (DFE)
Description	A one- to three-character file name extension to use for data files.
Valid Values	*ext* where *ext* is the name of the one- to three-character file name extension. This value is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option causes an error.

In other SQL statements, such as Select or Insert, users can specify an extension other than the one specified for this connection option. The Data File Extension value is used when no extension is specified.

The file extension cannot be one the driver already uses, such as MDX or CDX.

Default DBF

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 866 [dBASE]
General tab on page 869 [FoxPro]

Database Directory

Attribute Database (DB) [dBASE]

Description The directory that contains the data files.

Valid Values *database_directory*

where *database_directory* is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

Default None

GUI Tab General tab on page 866

Database Name

Attribute Database (DB) [FoxPro]

Description The directory that contains the database container (.DBC) files.

Valid Values *database_directory*

where *database_directory* is the full path name of the directory and .DBC file that you want to use.

Default None

GUI Tab General tab on page 869

Description

Attribute Description (n/a)

Description 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 on page 866 [dBASE]
General tab on page 869 [FoxPro]

Extension Case

Attribute ExtensionCase (EC)

Description This option determines whether uppercase or lowercase file extensions are accepted.

Valid Values LOWER | UPPER

When set to UPPER, uppercase extensions are accepted.

When set to LOWER, lowercase extensions are accepted.

Default UPPER

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

File Open Cache

Attribute FileOpenCache (FOC)

Description The maximum number of used file handles to cache.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default 0 (No File Open Caching)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled and/or if your database character set is not Unicode (refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details). The value you specify must match the database character encoding and the system locale.

The Driver Manager checks for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

International Sort

Attribute IntlSort (IS)

Description Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values 0 | 1

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default 0 (Disabled)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Lock Compatibility

Attribute LockCompatibility (LCOMP)

Description The locking scheme the driver uses when locking records.

Valid Values Clipper | dBASE | Fox | Q+E | Q+EVirtual

- Clipper specifies Clipper-compatible locking.
- dBASE specifies Borland-compatible locking.
- Fox specifies FoxPro-compatible locking.
- Q+E specifies that locks be placed on the actual bytes occupied by the record. Only applications that use the dBASE driver can read and write to the database. Other applications are locked out of the table completely (they cannot even read other records). This locking is compatible with earlier versions of Q+E products.
- Q+EVirtual specifies that locks be placed on bytes beyond the physical end-of-file. Q+EVirtual is the same as Q+E except that other applications can open the table and read the data.

The advantage of using a Q+E locking scheme over dBASE locking is that, on Inserts and Updates, Q+E locks only individual index tags, while dBASE locks the entire index. The following values determine locking support as described:

If you are accessing a table with an application that uses the dBASE driver, your locking scheme does not have to match the Create Type. If you access a table with two applications, however, and only one uses the dBASE driver, set your locking scheme to match the other application. For example, you do not have to set this value to Fox to work with a FoxPro table. But if you are using a FoxPro application simultaneously with an application using the dBASE driver on the same set of tables, set this value to Fox to ensure that your data does not become corrupted.

Default dBASE

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Locking

Attribute Locking (LCK)

Description The level of locking for the database file.

Valid Values NONE | RECORD | FILE

- NONE offers the best performance, but is intended only for single-user environments. See "Locking" on page 87 for details.
- RECORD locks only the records affected by the statement.
- FILE locks all of the records in the table.

Default RECORD

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Use Long Names

Attribute UseLongNames (ULN)

Description Specifies whether to use long file names as table names.

Valid Values 0 | 1

If set to 1 (Enabled), the driver uses long file names as table names. The maximum table name length is specific to the environment in which you are running.

If set to 0 (Disabled), the driver does not long file names as table names.

Default 0 (Disabled)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Use Long Qualifiers

Attribute UseLongQualifiers (ULQ)

Description Determines whether the driver uses long path names.

Valid Values 0 | 1

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default 0 (Disabled)

GUI tab Advanced tab on page 866 [dBASE]
Advanced tab on page 870 [FoxPro]

Defining Index Attributes on Windows

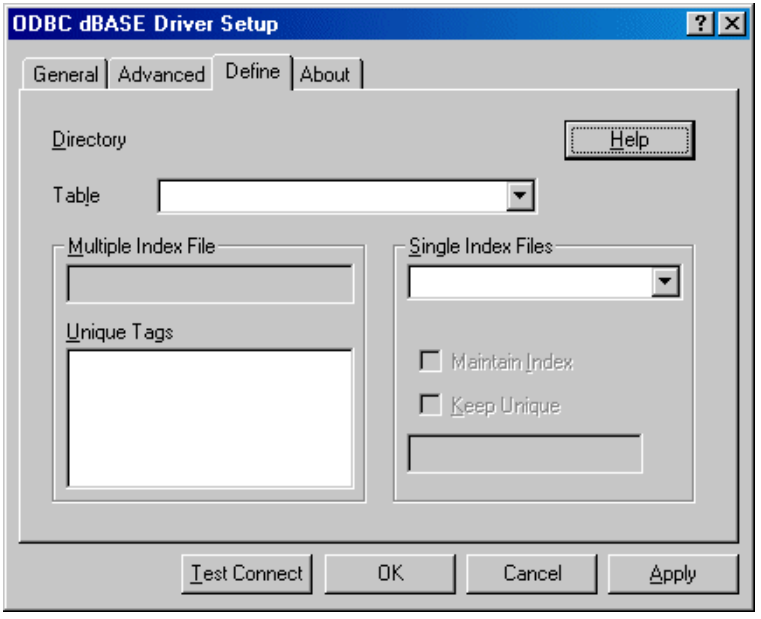
The Define tab of the ODBC dBASE Driver Setup dialog box allows you to define the attributes of index files. With the exception of Clipper, the family of databases that includes dBASE and FoxPro uses a multiple index file associated with a particular table (database file). This index file has a .MDX or .CDX extension and is automatically maintained by the driver. Tags within this index can be marked as unique.

Clipper uses single index files that are not automatically associated with a particular table. You can choose to have the driver maintain an index and choose whether or not the index is unique.

To define index file attributes:

1 Display the ODBC dBASE Driver Setup dialog box.

2 Click the **Define** tab.



On this tab, provide the following information; then, click **Apply**.

Table: Type or select the name of the table that contains the database information.

Multiple Index File: This field displays the name of any multiple index file (with a .CDX extension or .MDX extension) associated with the table you selected. This index file cannot be marked as unique, but tags within it can be.

Unique Tags: This field displays tags associated with the multiple index file. To mark tags as unique, click each one; each one remains selected until you click it again.

Single Index Files: The Single Index Files group is active only if you have selected a Clipper table.

Select the file from the drop-down list to define the attributes of a single index file.

Maintain Index: Select this check box to associate the specified single index file with the selected table.

Keep Unique: Select this check box to specify that the single index file is unique.

3 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Defining Index Attributes on UNIX and Linux

Index files for dBASE contain index tags for each index that exists for a database file. These index tags can be marked as unique, that is, the driver will ensure that no duplicate values exist for the columns that define the index tag. The unique attribute is not natively supported by the dBASE or FoxPro products. The enforcement and recognition of the unique attribute is an extension of the dBASE driver. The driver must be notified that index tags are unique. No configuration is needed for unique indexes that were created using the DataDirect Connect *for* ODBC dBASE driver. When using files that were not created with the dBASE driver, you must define unique index tags as outlined in the following procedure.

In the directory where the database and index files are located, use any text editor, such as vi, to define or edit the QEDBF.INI as follows:

1. Create a `[filename]` section where *`filename`* is the name of the database file. This entry is case-sensitive and the file extension must be included, for example, `[accts.dbf]`.
2. In the `[filename]` section, specify the number of unique indexes on the file (`NUMUNIQUE=`) and the index specifications (`UNIQUE#=index_filename,index_tag`). The *`index_tag`* can be determined by calling the ODBC function SQLStatistics and examining the INDEX_NAME result column.

For example, to define two unique indexes on the accts.dbf database file, the QEDBF.INI would be defined as:

```
[accts.dbf]
NUMUNIQUE=2
UNIQUE0=accts.mdx,ACCT_NAME
UNIQUE1=accts.mdx,ACCT_ID
```

Data Types

Table 17-2 shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement. Refer to Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference* for the syntax of the Create Table statement.

Table 17-3 shows how the additional FoxPro 3.0 data types map to the ODBC data types.

NOTE: A few products can create dBASE files with numbers that do not conform to the precision and scale of the Number column. For example, these products can store 100000 in a column declared as NUMBER(5,2). When this occurs, the dBASE driver displays error 1244, `Unsupported decimal format`. To remedy this situation, multiply the nonconforming column by 1, which converts it to the Float data type. For example:

```
SELECT BADCOL * 1 FROM BADFILE
```

`BADCOL * 1` is evaluated as an expression and is returned as a float value.

Table 17-2. dBASE Data Types

dBASE	ODBC
Binary [1]	SQL_LONGVARBINARY
Char [2]	SQL_CHAR
Date	SQL_TYPE_DATE
Float [3]	SQL_DECIMAL
General [4]	SQL_LONGVARBINARY
Logical	SQL_BIT

Table 17-2. dBASE Data Types (cont.)

Memo	SQL_LONGVARCHAR
Numeric	SQL_DECIMAL

1. dBASE V only.
2. 254 characters maximum (1024 for Clipper).
3. dBASE IV and V only.
4. FoxPro and dBASE V only.

Table 17-3. Additional FoxPro 3.0 Data Types

FoxPro 3.0	ODBC
Character (binary)	SQL_CHAR
Currency	SQL_DOUBLE
Datetime	SQL_TYPE_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
Memo (binary)	SQL_LONGVARBINARY

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Column Names

Column names in SQL statements (such as Select and Insert, for example) can be up to ten characters long. A column name can contain alphanumeric characters and the hyphen character (-). The first character must be a letter (a through z).

Select Statement

You use a SQL Select statement to specify the columns and records to be read. All of the Select statement clauses described in Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference* are supported by dBASE Select statements. This section describes the information that is specific to dBASE, which is Rowid.

Rowid Pseudo-Column

Each dBASE record contains a special column named Rowid. This field contains a unique number that indicates the record's sequence in the database. For example, a table that contains 50 records has Rowid values from 1 to 50 (if no records are marked deleted). You can use Rowid in Where and Select clauses.

Rowid is particularly useful when you are updating records. You can retrieve the Rowid of the records in the database along with the other field values. For example:

```
SELECT last_name, first_name, salary, rowid FROM emp
```

Then, you can use the Rowid of the record that you want to update to ensure that you are updating the correct record and no other. For example:

```
UPDATE emp set salary = 40000 FROM emp WHERE rowid=21
```

The fastest way of updating a single row is to use a Where clause with the Rowid. You cannot update the Rowid column.

Select statements that use the Rowid pseudo-column in the Where clause achieve maximum performance only for exact equality matches. If you use range scans instead of exact equality matches, a full table scan is performed. For example:

```
SELECT * FROM emp WHERE rowid=21    //fast search
SELECT * FROM emp WHERE rowid <=25  //full table scan
```

Alter Table Statement

The dBASE driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_name data_type |
ADD(column_name data_type [, column_name data_type]... ) |
DROP[COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept char (10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails if you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

Create and Drop Index Statements

The dBASE driver supports SQL statements to create and delete indexes.

Create Index

The type of index you create is determined by the value of the CreateType attribute, which you set in the driver Setup dialog box (for UNIX and Linux, edit the system information file) or as a connection string attribute. The index can be:

- dBASE IV or V (.MDX)
- Clipper (.NTX)
- FoxPro (.CDX)

The syntax for creating an index is:

```
CREATE [UNIQUE] INDEX index_name ON base_table_name
(field_name [ASC | DESC] [,field_name [ASC | DESC]]...)
```

Unique means that the driver creates an ANSI-style unique index over the column and ensures uniqueness of the keys. Use of unique indexes improves performance. ANSI-style unique indexes are different from dBASE-style unique indexes. With ANSI-style unique indexes, you receive an error message when

you try to insert a duplicate value into an indexed field. With dBASE-style unique indexes, you do not see an error message when you insert a duplicate value into an indexed field. This is because only one key is inserted in the index file.

index_name is the name of the index file. For FoxPro and dBASE IV or V, this is a tag, which is required to identify the indexes in an index file. Each index for a table must have a unique name.

base_table_name is the name of the database file whose index is to be created. The .DBF extension is not required; the driver automatically adds it if it is not present. By default, dBASE IV or V index files are named *base_table_name*.MDX and FoxPro indexes are named *base_table_name*.CDX.

field_name is a name of a column in the dBASE table. You can substitute a valid dBASE-style index expression for the list of field names.

ASC tells dBASE to create the index in ascending order. DESC tells dBASE to create the index in descending order. By default, indexes are created in ascending order. You cannot specify both ASC and DESC orders within a single Create Index statement. For example, the following statement is invalid:

```
CREATE INDEX emp_i ON emp (last_name ASC, emp_id DESC)
```

Table 17-4 shows the attributes of the different index files supported by the dBASE driver. For each type supported, it provides the following details:

- Whether dBASE-style unique indexes are supported
- Whether descending order is supported
- The maximum size supported for key columns
- The maximum size supported for the column specification in the Create Index statement
- Whether production/structural indexes are supported

Table 17-4. dBASE-Compatible Index Summary

Create Type .Extension	dBASE UNIQUE	DESC	Max Size of Key Column	Max Size of Column Specification	Production/ Structural Indexes	Supports FOR Expressions
dBASE IV, V .MDX	Yes	Yes	100	220	Yes	Yes
Clipper .NTX	Yes	Yes	250	255	No	Yes
FoxPro .IDX [1]	Yes	Yes	240	255	No	Yes
FoxPro .CDX	Yes	Yes	240	255	Yes	Yes

1. Compact IDX indexes have the same internal structure as a tag in a CDX file. These indexes can be created if the IDX extension is included with the index name in the Create Index statement.

Drop Index

The syntax for dropping an index is as follows:

```
DROP INDEX table_name.index_name
```

table_name is the name of the dBASE file without the extension.

For FoxPro and dBASE IV or V, *index_name* is the tag. Otherwise, *index_name* is the name of the index file without the extension.

To drop the index EMPHIRE.MDX, issue the following statement:

```
DROP INDEX emp.emphire
```

Pack Statement

When records are deleted from a dBASE file, they are not removed from the file. Instead, they are marked as having been deleted. Also, when memo fields are updated, space may be wasted in the files. To remove the deleted records and free the unused space from updated memo fields, you must use the Pack statement. It has the following form:

```
PACK filename
```

filename is the name of the dBASE file to be packed. The .DBF extension is not required; the driver automatically adds the extension if it is not present. For example:

```
PACK emp
```

You cannot pack a file that is opened by another user, and you cannot use the Pack statement in manual commit mode.

For the specified file, the Pack statement performs the following actions:

- Removes all deleted records from the file
- Removes the entries for all deleted records from .CDX and .MDX files having the same name as the file
- Compresses unused space in the memo file (.DBT or .FPT)

SQL Statements for FoxPro 3.0 Database Containers

The FoxPro DBC driver supports four additional SQL statements:

- Create Database
- Add Table
- Remove Table
- Use

To create a new FoxPro 3.0 database container, use:

```
CREATE DATABASE database_name
```

To add an existing table to the database container, use:

```
ADD TABLE table_name
```

To remove a table from the database container (not delete the table, but unlink it from the database container), use:

```
REMOVE TABLE table_name
```

To set the current database container to an existing database container, use:

```
USE database_name
```

To add or delete columns from a table in a database container, use the Alter Table statement (see "Alter Table Statement" on page 890).

Locking

With the dBASE driver, you can build and run applications that share dBASE database files on a network. Whenever more than one user is running an application that accesses a shared database file, the applications should lock the records that are being changed. Locking a record prevents other users from locking, updating, or deleting the record.

Levels of Database Locking

The dBASE driver supports three levels of database locking: NONE, RECORD, and FILE. You can set these levels in:

- The connection string (LCK=)
- The Setup dialog box

No locking offers the best performance, but is intended only for single-user environments.

With record or file locking, the system locks the database files during Insert, Update, Delete, or Select...For Update statements. The locks are released when the user commits the transaction. The locks prevent other users from modifying the locked objects, but they do not lock out readers.

With record locking, only records affected by the statement are locked. Record locking provides better concurrency with other users who also want to modify the database file.

With file locking, all the records in the database file are locked. File locking has lower overhead and may work better if records are modified infrequently, if records are modified primarily by one user, or if a large number of records are modified.

Limit on Number of Locks

There is a limit on the number of locks that can be placed on a file. If you are accessing a dBASE file from a server, the limit depends on the server (refer to your server documentation).

If you are accessing a dBASE file locally, the limit depends on the buffer space allocated when SHARE.EXE was loaded (refer to your DOS documentation). If you are exceeding the number of locks available, you may want to switch to file locking.

How Transactions Affect Record Locks

When an Update or Delete statement is run, the driver locks the records affected by that statement. The locks are released after the driver commits the changes. Under manual commit mode, the locks are held until the application commits the transaction. Under autocommit mode, the locks are held until the statement is run.

When a Select...For Update statement is run, the driver locks a record only when the record is fetched. If the record is updated, the driver holds the lock until the changes are committed. Otherwise, the lock is released when the next record is fetched.

Isolation and Lock Levels Supported

dBASE supports isolation level 1 (read committed). It supports both file-level and record-level locking.

Refer to Chapter 7 “Locking and Isolation Levels” in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The dBASE driver supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the dBASE driver. In addition, the following function is supported: SQLSetPos.

Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

18 The Informix Driver

The DataDirect Connect *for* ODBC Informix driver (the Informix driver) supports multiple connections to the following Informix database servers when using the appropriate client software.

Informix Dynamic Server 9.2*x,* 9.3*x*, 9.4*x,* 10, and 11

The Informix driver is 32-bit only and is supported in the Windows and UNIX environments, but not in the Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect product for the file name of the Informix driver.

NOTE: The Informix driver requires Informix client software. DataDirect Technologies also provides an Informix driver that does not require any client software to access Informix databases. See "The Informix Wire Protocol Driver" on page 309 for details.

Driver Requirements

This section provides the system requirements for using the Informix driver on all supported platforms.

Windows

To access supported remote Informix databases through the Informix driver, you need one of the following:

- Informix Connect for Windows platforms, version 2.*x*
- Informix Client Software Development Kit for Windows platforms, version 2.*x*

Use the Setnet32 utility supplied by Informix to define servers and the location of the INFORMIX directory. Use Ilogin to test your connection to the Informix server. The path to the ISQLT09A.DLL must be in your PATH environment variable.

UNIX (AIX, HP-UX PA-RISC, and Solaris)

The environment variable INFORMIXDIR must be set to the directory where you have installed the Informix client.

For example, the following syntax is valid for C-shell users:

```
setenv INFORMIXDIR /databases/informix
```

For Bourne- or Korn-shell users, the following syntax is valid:

```
INFORMIXDIR=/databases/informix;export INFORMIXDIR
```

In addition, the INFORMIXSERVER variable must be set to the name of the Informix server (as defined in your $INFORMIXDIR/etc/sqlhosts file). For further details, refer to the Informix documentation.

To access supported remote Informix databases through the Informix driver, you need one of the following:

- On AIX: Informix Client Software Development Kit version 2.2 or higher; or Informix Connect version 2.2 or higher
- On HP-UX and Solaris: Informix Connect version 2.*x*
- On HP-UX and Solaris: Informix Client Software Development Kit version 2.*x*

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 908 and Table 18-1 on page 912 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See "Environment Configuration" on page 45 for basic setup information and "Environment Variables" on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See "Configuration Through the Administrator" on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 18-1 on page 912 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure an Informix data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

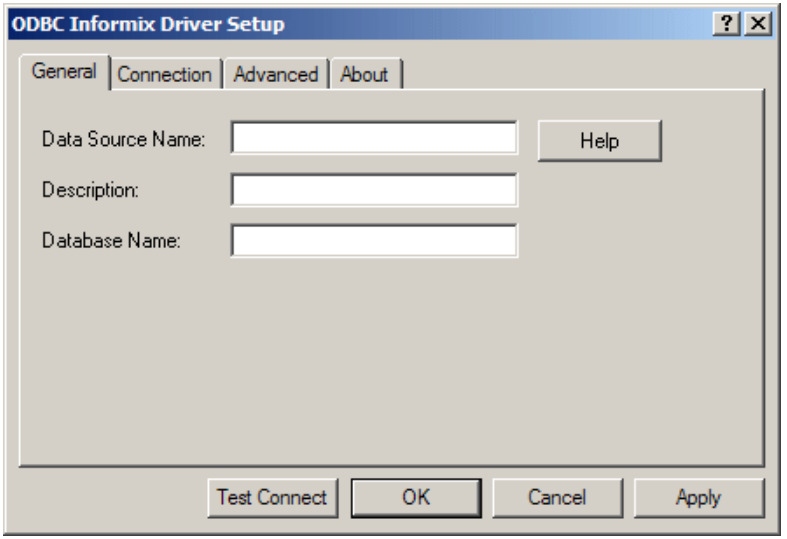
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

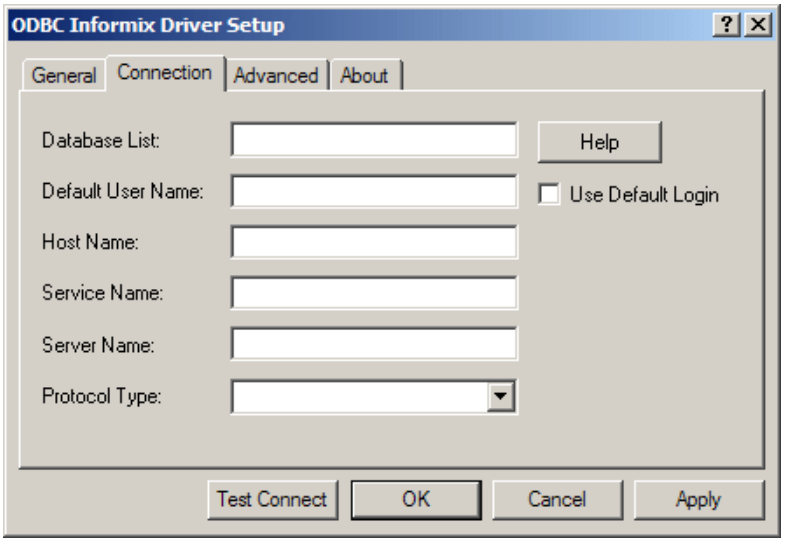


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 916)	None
Description (see page 917)	None
Database Name (see page 916)	None

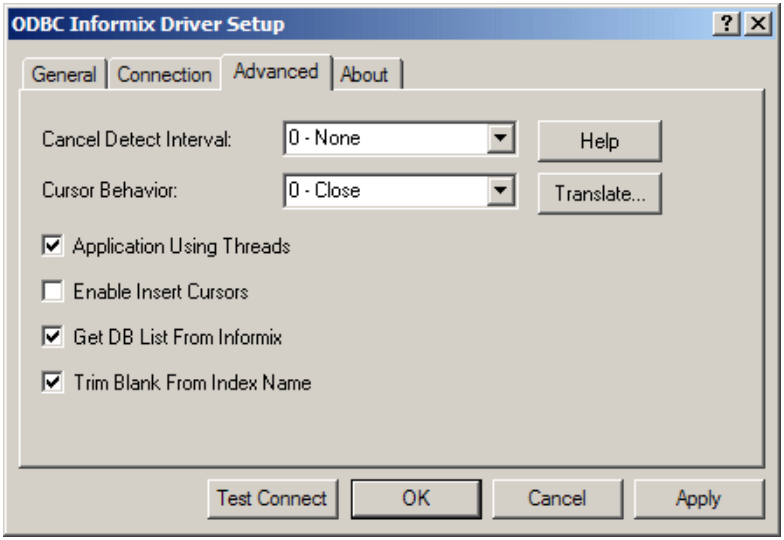
4 Optionally, click the **Connection** tab to specify connection information. If you want to configure the data source so that the logon dialog box does not appear during connection, you must specify the connection information on this tab.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Connection	Default
Database List (see page 915)	None
Default User Name (see page 916)	None
Use Default Login (see page 922) WINDOWS ONLY	Disabled
Host Name (see page 918)	None
Service Name (see page 921) WINDOWS ONLY	None
Server Name (see page 920)	None
Protocol Type (see page 920) WINDOWS ONLY	None

5 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Cancel Detect Interval (see page 914)	0 - None
Cursor Behavior (see page 915)	0 - Close
Application Using Threads (see page 913)	Enabled
Enable Insert Cursors (see page 917)	Disabled
Get DB List From Informix (GDBLFI) (see page 918)	Enabled
Trim Blank From Index Name (see page 921)	Enabled
IANAAppCodePage (see page 919) UNIX ONLY	4 (ISO 8559-1 Latin 1)

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies

provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 910 for details). The information you enter in the logon dialog box during a test connect is not saved.

- If the driver can connect, it releases the connection and displays a "connection established" message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message.

  Verify that all required client software is properly installed. If it is not, you will see the message:

  ```
  Specified driver could not be loaded due to system
  error [xxx].
  ```

  Click **OK**.

7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the

DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 18-1 on page 912 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 Informix is:

```
DSN=INFORMIX TABLES;DB=PAYROLL
```

A FILEDSN connection string is similar except for the initial keyword:

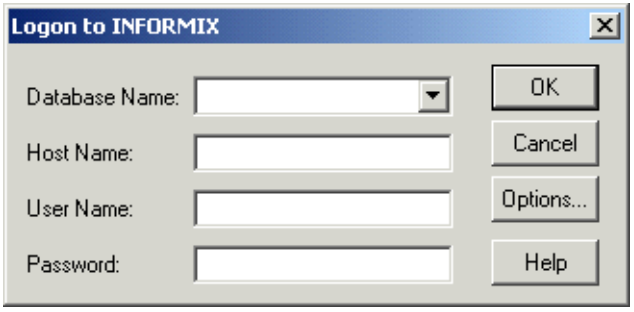
```
FILEDSN=Informix.dsn;DB=DBPAYROLL
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 Informix;
DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

Using a Logon Dialog Box

Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified.

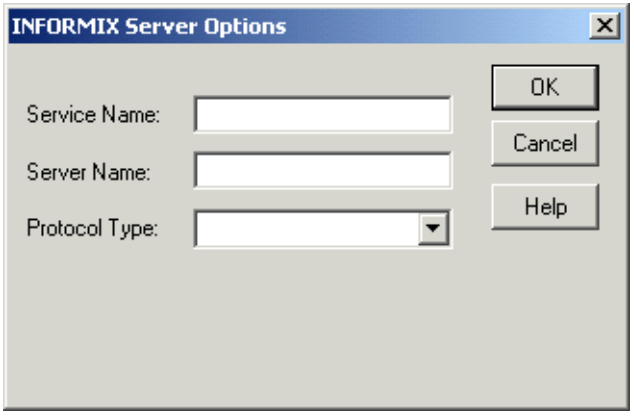


In this dialog box, provide the following information:

1 Type the name of the database you want to access, or, on Windows, select the name from the Database Name drop-down list.

On Windows, the names on the list are determined by the status of the **Get DB List From Informix** check box on the Advanced tab of the ODBC Informix driver Setup dialog box. If the check box is selected, the names displayed are returned from the Informix server. If cleared, the names displayed are returned from the user-entered list, which you specify in the Database List field on the Connection tab of the driver Setup dialog box.

2. Type the name of the host machine on which the Informix server resides.
3. If required, type your user name as specified on the Informix server.
4. If required, type your password.
5. On Windows, click **Options** to display the Informix Server Options dialog box, where you can change the Service Name, Server Name, and Protocol Type that you specified in the ODBC Informix Driver Setup dialog box. Click **OK** to save your changes.



6. Click **OK** to complete the logon and to update these values in the Registry.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 18-1 lists the connection string attributes supported by the Informix driver.

Table 18-1. Informix Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
CancelDetectInterval (CDI)	0 (None)
Database (DB)	None
Database (DL)	None
DataSourceName (DSN)	None
Description (n/a)	None
HostName (HOST)	None
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin 1)
LogonID (UID)	None

Table 18-1. Informix Attribute Names (cont.)

Attribute (Short Name)	Default
Password (PWD)	None
ServerName (SRVR)	None
TrimBlankFromIndexName (TBFIN)	1 (Enabled)
UseDefaultLogin (UDL)	0 (Disabled)

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

This connection option can affect performance. See "Performance Considerations" on page 922 for details.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI Tab Advanced tab on page 907

Cancel Detect Interval

Attribute	CancelDetectInterval (CDI)
Description	Determines whether long-running queries in threaded applications can be cancelled if the application issues a SQLCancel. This connection option can affect performance. See "Performance Considerations" on page 922 for details.
Valid Values	0 | *x* where *x* is the number of seconds the driver waits before checking for SQLCancel calls. If set to 0 (None), the driver does not allow long-running queries in threaded applications to be canceled, even if the application issues a SQLCancel. If set to *x* (seconds), for every pending query, the driver checks for SQLCancel calls at the specified interval. If the driver determines that a SQLCancel has been issued, the driver cancels the query.
Example	If you specify 5, for every pending query, the driver checks every five seconds to see whether the application has issued a SQLCancel call. If it detects a SQLCancel call, the driver cancels the query.
Default	0 (None)
GUI Tab	Advanced tab on page 907

Cursor Behavior

Attribute CursorBehavior (CB)

Description Determines whether cursors will be preserved or closed at the end of transactions.

Valid Values 0 | 1

If set to 1 (Enabled), cursors are held at their current position when transactions end. This value may slow the performance of your database operations.

If set to set to 0 (Disabled), cursors are closed at the end of transactions.

Default 0 (Disabled)

GUI Tab Advanced tab on page 907

Database List

Attribute Database (DL)

Description A list of database names that will be displayed in the Logon dialog box if **Get DB List From Informix** on the Advanced tab is *not* selected.

Valid Values *database_name*[,*database_name*][...]

where *database_name* is a database name you want to appear in the Logon dialog box. Separate multiple values with commas.

Example db1, db2, db3

Default None

GUI Tab Connection tab on page 906

Database Name

Attribute: Database (DB)

Description: The name of the database to which you want to connect.

Valid Values: *database_name*

where *database_name* is the name of a valid database.

Default: None

GUI Tab: General tab on page 905

Data Source Name

Attribute: DataSourceName (DSN)

Description: 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 on page 905

Default 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 Connection tab on page 906

Description

Attribute Description (n/a)

Description 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 on page 905

Enable Insert Cursors

Attribute EnableInsertCursors (EIC)

Description Determines whether the driver can use Insert cursors during inserts governed by parameters.

Valid Values If set to 1 (Enabled), the driver uses Insert cursors.

If set to 0 (Disabled), the driver does not use Insert cursors.

Default 1 (Enabled)

GUI Tab Advanced tab on page 907

Get DB List From Informix (GDBLFI)

Attribute: GetDBListFromInformix (GDBLFI)

Description: Determines whether the driver requests the database list to be returned from the Informix server or from the database list that the user entered at driver setup.

Valid Values: 0 | 1

If set to 1 (Enabled), the driver requests the database list from the Informix server.

If set to 0 (Disabled), the driver uses the list that was entered by the user at driver setup.

Default: 1 (Enabled)

GUI Tab: Advanced tab on page 907

Host Name

Attribute: HostName (HOST)

Description: The name of the server to which you want to connect.

Valid Values: *server_name*

where *server_name* is the name of the server to which you want to connect.

Default: None

GUI Tab: General tab on page 905

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled and/or if your database character set is not Unicode (refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details). The value you specify must match the database character encoding and the system locale.

The Driver Manager checks for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 907

Password

Attribute: Password (PWD)

Description: The password that the application uses to connect to your database. The Password option cannot be specified through the driver Setup dialog box and should not be stored in a data source. It is specified through the Logon dialog box or a connection string.

Valid Values: *pwd*

where *pwd* is a valid password.

Default: None

GUI Tab: n/a

Protocol Type

Attribute: Protocol (PRO)

Description: Determines the protocol used by the driver to communicate with the server.

Valid Values: olsocspx | olsoctcp | onsocspx | onsoctcp | seipcpip | sesocspx | sesoctcp

Specify the appropriate Informix protocol.

Default: None

GUI Tab: Connection tab on page 906

Server Name

Attribute: ServerName (SRVR)

Description: The name of the Informix server.

Valid Values *server_name*

where *server_name* is a name that uniquely identifies the Informix server.

Default None

GUI Tab Connection tab on page 906

Service Name

Attribute Service (SERV)

Description The name of the Informix service. The service name is assigned by the system administrator.

Valid Values *service_name*

where *service_name* is the a name that uniquely identifies the Informix service. This name must be specified as it appears in the services file on the server machine.

Default None

GUI Tab Connection tab on page 906

Trim Blank From Index Name

Attribute TrimBlankFromIndexName (TBFIN)

Description Determines whether the driver trims leading spaces from system-generated index names. Some applications cannot process a leading space in index names.

Valid Values If set to 1 (Enabled), the driver trims leading spaces from system-generated index names.

If set to 0 (Disabled), the driver does not trim leading spaces from system-generated index names.

Default 1 (Enabled)

GUI Tab Advanced tab on page 907

Use Default Login

Attribute UseDefaultLogin (UDL)

Description Determines where the driver reads login credentials.

Valid Values 0 | 1

If set to 0 (Disabled), login credentials are read from the Windows Registry, the connection string, or the Logon to Informix dialog box.

If set to 1 (Enabled), login credentials are read directly from the Informix registry.

Default 0 (Disabled)

GUI Tab Connection tab on page 906

Performance Considerations

The following connection options can enhance driver performance. You can also enhance performance through efficient application design. Refer to Chapter 5 "Designing ODBC Applications for Performance Optimization" in the *DataDirect Connect Series for ODBC Reference* 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.

Cancel Detect Interval (CancelDetectInterval): If your application uses threads, it may allow canceling of long running queries (may issue synchronous SQLCancel calls). If your application does not issue synchronous SQLCancel calls, the driver can improve performance if Cancel Detect Interval is disabled (set to 0). In this case, the driver does not incur the overhead of periodically checking for SQLCancel. In the case where your application does issue synchronous SQLCancel calls, this option should be set to a value that specifies how often the driver checks to see if a long running query has been canceled.

Data Types

Table 18-2 shows how the Informix data types map to the standard ODBC data types.

Table 18-2. Informix Data Types

Informix	ODBC
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
BYTE [1]	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB	SQL_LONGVARCHAR
DATE	SQL_TYPE_DATE
DATETIME YEAR TO FRACTION(f) [2]	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO SECOND	SQL_TYPE_TIMESTAMP
DATETIME YEAR TO DAY	SQL_TYPE_DATE
DATETIME HOUR TO SECOND	SQL_TYPE_TIME
DATETIME HOUR TO FRACTION(f) [2]	SQL_TYPE_TIME

Table 18-2. Informix Data Types (cont.)

Informix	ODBC
DECIMAL	SQL_DECIMAL
FLOAT	SQL_DOUBLE
INT8	SQL_BIGINT
INTEGER	SQL_INTEGER
INTERVAL YEAR(p) TO YEAR	SQL_INTERVAL_YEAR
INTERVAL YEAR(p) TO MONTH	SQL_INTERVAL_YEAR_TO_MONTH
INTERVAL MONTH(p) TO MONTH	SQL_INTERVAL_MONTH
INTERVAL DAY(p) TO DAY	SQL_INTERVAL_DAY
INTERVAL DAY(p) TO HOUR	SQL_INTERVAL_DAY_TO_HOUR
INTERVAL DAY(p) TO MINUTE	SQL_INTERVAL_DAY_TO_MINUTE
INTERVAL DAY(p) TO SECOND	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL DAY(p) TO FRACTION(f) [2]	SQL_INTERVAL_DAY_TO_SECOND
INTERVAL HOUR(p) TO HOUR	SQL_INTERVAL_HOUR
INTERVAL HOUR(p) TO MINUTE	SQL_INTERVAL_HOUR_TO_MINUTE
INTERVAL HOUR(p) TO SECOND	SQL_INTERVAL_HOUR_TO_SECOND
INTERVAL HOUR(p) TO FRACTION(f) [2]	SQL_INTERVAL_HOUR_TO_SECOND
INTERVAL MINUTE(p) TO MINUTE	SQL_INTERVAL_MINUTE
INTERVAL MINUTE(p) TO SECOND	SQL_INTERVAL_MINUTE_TO_SECOND
INTERVAL MINUTE(p) TO FRACTION(f) [2]	SQL_INTERVAL_MINUTE_TO_SECOND
INTERVAL SECOND(p) TO SECOND	SQL_INTERVAL_SECOND
INTERVAL SECOND(p) TO FRACTION(f) [2]	SQL_INTERVAL_SECOND
LVARCHAR(p) [2]	SQL_VARCHAR
MONEY	SQL_DECIMAL
NCHAR	SQL_CHAR
NVARCHAR	SQL_VARCHAR
SERIAL	SQL_INTEGER
SERIAL8	SQL_BIGINT
SMALLFLOAT	SQL_REAL
SMALLINT	SQL_SMALLINT

Table 18-2. Informix Data Types (cont.)

Informix	ODBC
TEXT [1]	SQL_LONGVARCHAR
VARCHAR [1]	SQL_VARCHAR

1. Not supported for Standard Engine databases.
2. (f) can have a value of 1, 2, 3, 4, or 5. The precision is type-dependent and the scale is 5.

The Informix driver does not support any complex data types (for example, set, multiset, list, and named/unnamed abstract types). When the driver encounters a complex type it will return an Unknown Data Type error (SQL State HY000).

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

MTS Support

On Windows, the driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC) using the XA Protocol. For a general discussion of MTS and DTC, refer to the help file of the Microsoft Transaction Server SDK.

NOTE: The DataDirect Connect *for* ODBC 32-bit drivers can operate in a 64-bit Windows environment; however, they do not support DTC in this environment. Only the DataDirect Connect64 *for* ODBC 64-bit drivers support DTC in a 64-bit Windows environment.

To enable DTC support, you must be using Informix Connect version 2.20 or higher clients.

To enable support for the DTC:

1 Use the **Setnet32** utility supplied by Informix to define:

- The INFORMIXDIR environment variable, which identifies the location of the client programs, library files, message files, header files, and other Informix software components
- The INFORMIXSERVER environment variable, which identifies the default database server
- An Informix server, which identifies either an existing Informix database server or a new one
- A host name, which identifies the host computer with the database server you want to use
- A user name, which identifies a user name for an account on the currently selected host computer
- A password for the specified user name, if required

When enlisting in a distributed transaction, the Informix clients only use the defaults specified in **Setnet32**.

2 Run the **regcopy** utility provided with INFORMIX-Connect to copy the registry entries created by **Setnet32** to an area in the registry that is accessible by the DTC. The DTC is a service, and services do not search for configuration information in the Windows registry where **Setnet32** stores client products environment variables. Therefore, if you do not run regcopy after setting the defaults in **Setnet32**, enlistment in a distributed transaction will fail.

For information on using the **Setnet32** and **regcopy** utilities, see the Informix documentation.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

Isolation and Lock Levels Supported

If connected to an Online Server, Informix supports isolation levels 0 (read uncommitted), 1 (read committed), and 3 (serializable). The default is 1. The Standard Engine supports isolation level 0 (read uncommitted) only.

Informix also supports an alternative isolation level 1, called "cursor stability." Your ODBC application can use this isolation level by calling SQLSetConnectAttr (1040,1).

Additionally, if transaction logging has not been enabled for your database, then transactions are not supported by the driver (the driver is always in auto-commit mode).

Informix supports page-level and row-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

The following functions are supported:

- SQLProcedures
- SQLColumnPrivileges
- SQLTablePrivileges
- SQLPrimaryKeys
- SQLForeignKeys
- SQLProcedureColumns

The driver also supports scrollable cursors with SQLFetchScroll or SQLExtendedFetch. The driver supports the core SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Informix driver.

Number of Connections and Statements Supported

The Informix driver supports multiple connections and multiple statements per connection to the Informix database system.

19 The Paradox Driver

The DataDirect Connect *for* ODBC Paradox driver (the Paradox driver) supports:

Paradox 4, 5, 7, 8, 9, and 10 database files

The Paradox driver is 32-bit only and is supported in the Windows environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the Paradox driver.

Driver Requirements

To use the Paradox driver, you must install the Borland Database Engine, which conforms to the IDAPI programming interface. The Borland Database Engine can be found in any of the following software packages:

- Borland C++ for Windows
- Delphi for Windows
- Paradox 4, 5, 7, 8, 9, or 10 for Windows

You must have IDAPI32.DLL on your path; in your Windows 98 or Windows Me \SYSTEM directory; or in your Windows NT, Windows 2000, Windows XP, Windows Vista, or Windows Server 2003 \SYSTEM32 directory.

Multiuser Access to Database Files

You can query Paradox database files that reside in a shared directory on a network or that are to be shared among applications running on a local workstation. If the database files are on a network server, multiple users can query these database files simultaneously. To share Paradox database files among multiple users, the database files must be located in a shared directory on your network server.

Two connection string attributes identify the Paradox database you are accessing: Database (database directory) and NetDir (network directory). The Database setting specifies the directory in which the Paradox database files are stored. If the Database setting you specify is a shared network directory, then Paradox requires a NetDir specification as well. This value identifies the directory containing the PARADOX.NET file that corresponds to the Database setting you have specified.

Every user who accesses the same shared directory of database files must set the NetDir value to point to the same PARADOX.NET file. If your connection string does not specify a NetDir value, then Paradox uses the NetDir value specified in the Paradox section of the IDAPI configuration file. This makes it important that the NetDir specification in each user's IDAPI configuration file be set correctly.

Whenever you open a Paradox database file that another user opens at the same time, the consistency of the data becomes an issue if both individuals are updating the database file.

Locking

The Paradox driver has two locking levels: record locking and table locking. Database files that have no primary key always have a prevent write lock placed on the database file when the

file is opened. The table lock is escalated to a write lock when an operation that changes the database file is attempted.

Database files that have primary keys use record locking. The locking level is escalated from record locking to a table write lock if the transaction runs out of record locks.

Primary keys provide the greatest concurrency for database files that are accessed and modified by multiple users.

NOTE: If a table lock is placed on a Paradox database file, the Paradox driver prevents users from updating and deleting records but does not prevent them from inserting records into the locked database file.

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 935 and Table 19-1 on page 937 for an alphabetical list of driver connection string attributes and their initial default values.

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Paradox data source:

1 Start the ODBC Administrator by selecting its icon from the DataDirect Connect program group; then, select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name on the User DSN tab and click **Configure** to display the driver Setup dialog box.

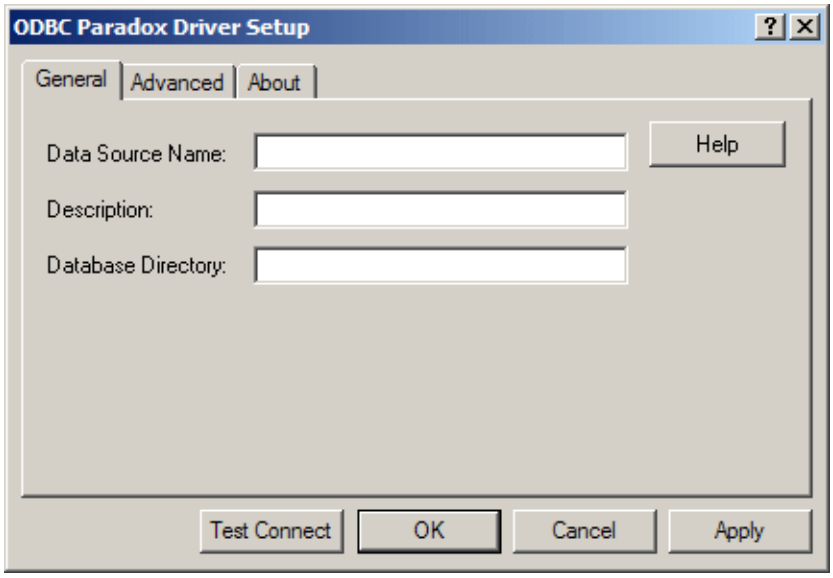
  If you are configuring a new user data source, click **Add** on the User DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **System DSN**: To configure a new system data source, click **Add** on the System DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source name on the File DSN tab and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** on the File DSN tab to display a list of installed drivers. Select the driver and click **Next**. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

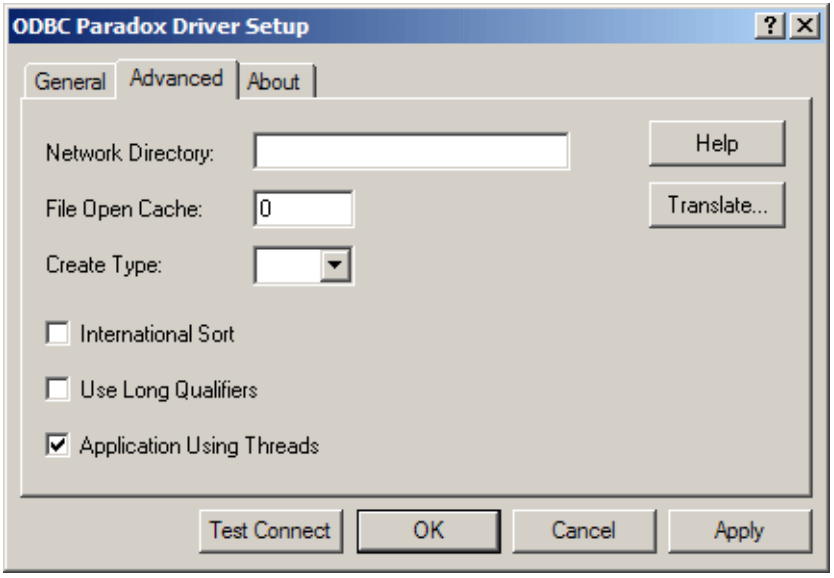


NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

2 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 939)	None
Description (see page 940)	None
Database Directory (see page 940)	None

3 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Network Directory (see page 942)	None
File Open Cache (see page 940)	0
Create Type (see page 938)	None
International Sort (see page 941)	Disabled
Use Long Qualifiers (see page 942)	Disabled
Application Using Threads (see page 938)	Enabled

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

4 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box. 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.

     Verify that all required client software is properly installed. If it is not, you will see the message:

     ```
     Specified driver could not be loaded due to system
     error [xxx].
     ```

     Click **OK**.

5 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=

value pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 19-1 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 Paradox is:

```
DSN=PARADOXDBFILES;DB=C:\ODBC\EMP;DQ=0
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Paradox.dsn;DB=C:\ODBC\EMP;DQ=0
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 ParadoxFile (*.db);DB=C:\ODBC\EMP
```

Connection Options Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 19-1 lists the connection string attributes supported by the Paradox driver.

Table 19-1. Paradox Attribute Names

Attribute (Short Name)	Default
ApplicationUsingThreads (AUT)	1 (Enabled)
Create Type (CT)	None
Database (DB)	None
DataSourceName (DSN)	None
Description (n/a)	None
FileOpenCache (FOC)	0 (No File Open Caching)
IntlSort (IS)	0 (Disabled)

Table 19-1. Paradox Attribute Names (cont.)

Attribute (Short Name)	Default
NetDir (ND)	None
UseLongQualifiers (ULQ)	0 (Disabled)

Application Using Threads

Attribute: ApplicationUsingThreads (AUT)

Description: Determines whether the driver works with applications using multiple ODBC threads.

Valid Values: 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default: 1 (Enabled)

GUI tab: Advanced tab on page 934

Create Type

Attribute: Create Type (CT)

Description: The Paradox database file version used for any Create Table statements that you execute.

Valid Values 3 | 4 | 5 | 7 | 8 | 9 | 10 | (null)

The numeric values map to the major revision numbers of the Paradox family of products. If you do not specify a version, the default version is determined by the Level setting in the Paradox section of the IDAPI configuration file.

To override another CreateType setting chosen during data source configuration with the default create type determined by the Level setting in the Paradox section of the IDAPI configuration file, set CreateType= (null).

NOTE: When CreateType is set to 7, 8, 9 or 10, the Paradox driver supports database file names to maximum of 128 characters. For all other CreateType settings, the driver supports database file names to a maximum of eight characters.

Default None

GUI tab Advanced tab on page 934

Data Source Name

Attribute DataSourceName (DSN)

Description 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 on page 933

Database Directory

Attribute: Database (DB)

Description: The directory that contains the data files.

Valid Values: *database_directory*

where *database_directory* is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

If you have an IDAPI configuration file, you can specify aliases that are defined in this file. Enclose the alias name within colons. For example, to use the alias MYDATA, specify ":MYDATA:".

Default: None

GUI Tab: General tab on page 933

Description

Attribute: Description (n/a)

Description: 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 on page 933

File Open Cache

Attribute: FileOpenCache (FOC)

Description: The maximum number of used file handles to cache.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default 0 (No File Open Caching)

GUI tab Advanced tab on page 934

International Sort

Attribute IntlSort (IS)

Description Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values 0 | 1

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default 0 (Disabled)

GUI tab Advanced tab on page 934

Network Directory

Attribute NetDir (ND)

Description The directory containing the PARADOX.NET file that corresponds to the database you have specified. If the Paradox database you are using is shared on a network, every user who accesses it must set this value to point to the same PARADOX.NET file.

Valid Values *network_directory*

where *network_directory* is the full path name of the directory containing the PARADOX.NET file.

If a directory is not specified, this value is determined by the NetDir setting in the Paradox section of the IDAPI configuration file. If you are not sure how to set this value, contact your network administrator.

Default None

GUI tab Advanced tab on page 934

Use Long Qualifiers

Attribute UseLongQualifiers (ULQ)

Description Determines whether the driver uses long path names.

Valid Values 0 | 1

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default 0 (Disabled)

GUI tab Advanced tab on page 934

Data Types

Table 19-2 shows how the Paradox data types are mapped to the standard ODBC data types. It also identifies the types supported by Paradox versions 4.*x*, 5.*x*, and higher. These Paradox data types can be used in a Create Table statement. See "Create Table Statement" on page 946 for the syntax of the Create Table statement.

Table 19-2. Paradox Data Types

Paradox	ODBC	4.*x* Support	5.*x* and Higher Support
Alpha	SQL_CHAR	Yes	Yes
AutoIncrement	SQL_INTEGER	No	Yes
BCD	SQL_DECIMAL	No	Yes
Binary [1]	SQL_LONGVARBINARY	Yes	Yes
Bytes [1]	SQL_BINARY	No	Yes
Date	SQL_TYPE_DATE	Yes	Yes
Formatted Memo [1]	SQL_LONGVARBINARY	Yes	Yes
Graphic [1]	SQL_LONGVARBINARY	Yes	Yes
Logical [1]	SQL_BIT	No	Yes
Long Integer	SQL_INTEGER	No	Yes
Memo [1]	SQL_LONGVARCHAR	Yes	Yes
Money	SQL_DOUBLE	Yes	Yes
Number	SQL_DOUBLE	Yes	Yes
OLE [1]	SQL_LONGVARBINARY	Yes	Yes
Short	SQL_SMALLINT	No	Yes

Table 19-2. Paradox Data Types (cont.)

Paradox	ODBC	4.*x* Support	5.*x* and Higher Support
Time	SQL_TYPE_TIME	No	Yes
TimeStamp	SQL_TYPE_TIMESTAMP	No	Yes

1. Cannot be used in an index. Of these types, only Logical can be used in a Where clause.

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Select Statement

You use a SQL Select statement to specify the columns and records to be read. All of the Select statement clauses described in Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference* are supported by the Paradox driver. This section describes the information that is specific to the Paradox driver.

Column Names

Paradox column names are case-insensitive and their maximum length is 25 characters. If a column name contains a special character, does not begin with a letter, or is a reserved word, surround it with the grave character (`) (ASCII 96). For example:

```
SELECT `last name` FROM emp
```

Alter Table Statement

The Paradox driver supports the Alter Table statement to add one or more columns to a Paradox database file or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE filename {ADD column_name data_type
[DEFAULT default_value] | ADD (column_name data_type
[DEFAULT default_value][, column_name data_type
[DEFAULT default_value]] , , ,) | DROP
[COLUMN] column_name [CASCADE | RESTRICT]}
```

filename is the name of the Paradox database file to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp database file:

```
ALTER TABLE emp (ADD startdate date, dept alphanumeric(10))
```

Dropping Columns

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time.

When dropping a column, use the Cascade keyword to drop the column while removing references from any dependent objects, such as indexes or views. Use Restrict to cause the Alter Table statement to fail if other objects are dependent upon the column you are dropping. For example, to drop a column and remove its references from dependent objects:

```
ALTER TABLE emp DROP startdate CASCADE
```

If the Alter Table statement contains neither Cascade nor Restrict, it fails when you attempt to drop a column upon which other objects are dependent.

Create Table Statement

The Create Table statement is used to create database files. The form of the Create Table statement is:

```
CREATE TABLE filename (col_definition[,col_definition,...])
```

filename can be a simple name or a full name. A simple file name is preferred for portability to other SQL data sources. If it is a simple file name, the file is created in the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is created in the directory you specified as the database directory in the Registry. If you did not specify a database directory in either place, the file is created in the current working directory at the time you connected to the driver.

col_definition is the column name, followed by the data type, Default clause, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is Not Null. Not all flat-file database files support Not Null columns. In the cases where Not Null is not supported, this restriction is ignored and the driver returns a warning if Not Null is specified for a column. The Not Null column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a Not Null restriction).

A sample Create Table statement to create a Paradox database file named emp is:

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL DEFAULT
'JOHNSON', first_name CHAR(12) NOT NULL,
salary NUMERIC (10,2) NOT NULL, hire_date DATE NOT NULL)
```

Password Protection

Paradox supports two types of passwords: master and auxiliary. The Paradox driver supports master passwords only and can manage a maximum of 50 passwords.

Paradox database files can be encrypted to provide limited access to users who do not know the password. The driver maintains a list of passwords for each connection. The driver can access only encrypted database files for which a password appears in this list. You can supply a password in three ways: by typing it in the Password dialog box (which appears when the driver does not have the password to open an encrypted database file), by including it in a connection string (with the Passwords attribute), or by using the Add Password statement.

Paradox provides five statements that manage passwords for Paradox database files. These statements are specific to the Paradox driver:

```
ENCRYPT filename USING PASSWORD password
ADD PASSWORD password
DECRYPT filename USING PASSWORD password
REMOVE PASSWORD password
REMOVE ALL PASSWORDS
```

filename can be a simple file name or a full path name. If a simple file name is given, the file must be in the directory specified with the Database connection string attribute. The .DB extension is not required.

password is a case-sensitive text string and its maximum length is 15 characters, including blanks. If your password includes lowercase letters or nonalphanumeric characters, enclose it in single quotation marks (').

Encrypting a Paradox Database File

The Encrypt statement associates a password with a database file. For example:

```
ENCRYPT emp USING PASSWORD test
```

Accessing an Encrypted Paradox Database File

To access an encrypted Paradox database file, add the password to the list of passwords Paradox maintains for that connection. To do so, you can:

- Issue an Add Password statement before you access the database file. For example:

  ```
  ADD PASSWORD test
  SELECT * FROM emp
  ```

- Specify the passwords using the Passwords attribute at connection time.

If you do not add the password, the driver prompts you for it when you access the database file.

Decrypting a Paradox Database File

The Decrypt statement disassociates a password from a database file. You do not need to enter the password to open the database file. For example:

```
DECRYPT emp USING PASSWORD test
```

Removing a Password from Paradox

The Remove Password statement removes a password from the list Paradox maintains for the connection. For example:

```
REMOVE PASSWORD test
```

Removing All Passwords from Paradox

The Remove All Passwords statement removes the list of passwords Paradox maintains.

If you remove a password from Paradox and do not decrypt the database file, you must continue entering the password to open the database file.

Index Files

An index is used to read records in sorted order and to improve performance when selecting records and joining database files. Paradox indexes are stored in separate files and are either *primary* or *non-primary*.

Primary Index

A primary index is made up of one or more fields from the Paradox database file. The primary key fields of a primary index consist of one or more consecutive fields in the database file, beginning with the first field in the database file. A database file can have only one primary index.

Collectively, the primary key fields uniquely identify each record in the Paradox database file. Thus, no two records in a Paradox database file can share the same values in their primary key fields. Once a primary index is created for a Paradox database file, the database file's records are re-ordered based on the primary key fields. At the time a primary index is created, if any records have matching primary key field values, Paradox deletes all but the first record. Paradox creates this index as maintained; that is, if you modify, add, or delete records in the database file, the primary index is updated automatically to reflect these changes.

A primary index is a single file with the same name as the database file on which it is based but with a .PX extension.

To lock records, you must have a primary index.

Non-Primary Index

Paradox 7, 8, 9, and 10 database files support UNIQUE secondary indexes. See "Create and Drop Index Statements" on page 951 for more information.

A non-primary index is defined by specifying one or more fields in the Paradox database file that constitute the non-primary key field. It allows Paradox to sort each record in the database file according to the values of the non-primary key fields.

There are two kinds of non-primary indexes: maintained and non-maintained. A maintained index is automatically updated

when the database file is changed, whereas a non-maintained index is not. Instead, a non-maintained index is tagged out-of-date and is updated when the index is used again.

You must have a primary index on a database file before you create a maintained, non-primary index.

The Paradox driver uses non-maintained indexes only for read-only queries on locked database files. A primary index is not required for the non-maintained index to be used.

For Paradox 3.*x*, a single non-primary index consists of a pair of files with the same name as the database file on which the non-primary index is based; one of these files has an .X*nn* extension while the other has a .Y*nn* extension (where the hexadecimal number *nn* corresponds to the field number of the non-primary key field for the non-primary index).

For Paradox 4.*x*, 5, 7, 8, 9, and 10 single-field, non-primary indexes that are case-sensitive have the same name as their associated database file and are assigned file extensions .X01 through .XFF, depending on the number of the field on which the index is based. Single-field, non-primary indexes that are case insensitive and composite indexes have the same name as the database file on which they are based. They are assigned file extensions sequentially starting with .XG0 (with hexadecimal increments).

Create and Drop Index Statements

The Paradox driver supports SQL statements to create and delete indexes. The Create Index Primary statement is used to create primary indexes. The Create Index statement is used to create non-primary indexes. The Drop Index statement is used to delete indexes.

Create Index Primary Statement

The syntax for creating a primary index is:

```
CREATE [UNIQUE] INDEX PRIMARY ON
filename (column [,column...])
```

The UNIQUE keyword is optional; the index is unique regardless of whether you include this keyword.

filename is the name of the Paradox database file on which the index is to be based.

column is the name of a column that is included as the key field for the index. The column list must contain one or more consecutive fields in the database file, beginning with the first field in the database file.

For example:

```
CREATE UNIQUE INDEX PRIMARY ON emp (emp_id)
```

Be careful when you create a primary key because any rows that have a primary key duplication are deleted when you execute the Create Index Primary statement.

Create Index Statement

For Paradox 4.0, 4.5, and 5.0 database files, the syntax for creating a non-primary index is:

```
CREATE INDEX index_name
[/NON_MAINTAINED] [/CASE_INSENSITIVE] ON
filename (column [, column...])
```

For Paradox 7, 8, 9, and 10 database files, the syntax for creating a non-primary index is:

```
CREATE [UNIQUE] INDEX index_name
[/NON_MAINTAINED] [/CASE_INSENSITIVE] ON
filename [column [DESC] [,column...]
```

For Paradox 7, 8, 9, and 10 database files only (when the Create Type is 7, 8, 9 or 10), the optional UNIQUE keyword prevents duplicate values in the non-primary index.

index_name identifies the index. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96).

The NON_MAINTAINED switch makes the index non-maintained. The default is to create a maintained index.

The CASE_INSENSITIVE switch makes the index case-insensitive. The default is to create a case-sensitive index.

filename is the name of the Paradox database file on which the index is to be based.

column is the name of a column that is included as the key field for the index.

For Paradox 7, 8, 9 or 10 database files only (when the Create Type is 7, 8, 9 or 10), the DESC keyword creates a non-primary index that uses descending keys.

Drop Index Statement

The syntax for dropping a primary index is:

```
DROP INDEX path_name.PRIMARY
```

For example:

```
DROP INDEX emp.PRIMARY
```

The syntax for dropping a non-primary index is:

```
DROP INDEX path_name.index_name
```

path_name is the name of the Paradox database file from which the index is being dropped. The pathname can be either the fully qualified pathname or, if the database file is specified with the Database attribute of the connection string, a simple database file name.

index_name is the name that was given to the index when it was created. If the name contains blanks or special characters, or does not begin with a letter, surround it with the grave character (`) (ASCII 96).

For example:

```
DROP INDEX emp.last_name
```

Transactions

The Paradox driver supports transactions. A transaction is a series of database changes that is treated as a single unit. In applications that do not use transactions, the Paradox driver immediately executes Insert, Update, and Delete statements on the database files and the changes are automatically committed when the SQL statement is executed. There is no way to undo such changes. In applications that use transactions, inserts, updates, and deletes are held until a Commit or Rollback is specified. A Commit saves the changes to the database file; a Rollback discards the changes.

Transactions affect the removal of record locking. All locks are removed when SQLTransact is called with the Commit or Rollback option to end the active transaction.

Isolation and Lock Levels Supported

Paradox supports isolation levels 1 (read committed) and 3, (serializable). It supports record- and table-level locking.

Refer to Chapter 7 "Locking and Isolation Levels" in the *DataDirect Connect Series for ODBC Reference* for details.

ODBC Conformance Level

When used with Paradox 5, 7, 8, 9, or 10 database files, the Paradox driver supports the SQLForeignKeys function. The Paradox driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The Paradox driver supports the minimum SQL grammar. In addition, the following functions are supported:

- SQLSetPos
- SQLPrimaryKeys

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Paradox driver.

Number of Connections and Statements Supported

The Paradox database system supports multiple connections and multiple statements per connection.

20 The Text Driver

The DataDirect Connect *for* ODBC Text driver (the Text driver) supports:

ASCII text files

These files can be printed directly or edited with text editors or word processors, because none of the data is stored in a binary format.

The Text driver is 32-bit only and is supported in the Windows, UNIX, and Linux environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

The Text driver executes SQL statements directly on the text files. The driver supports Insert statements and inserts the record at the end of the file. You can execute Update and Delete statements conditionally.

The Text driver can access files up to 15 GB in size.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the text driver.

Driver Requirements

There are no client requirements for the Text driver.

Formats for Text Files

Some common formats for text files are listed in Table 20-1.

Table 20-1. Common Text File Formats

Format	Description
Comma-separated values	Commas separate column values, and each line is a separate record. Column values can vary in length. These files often have the .CSV extension.
Tab-separated values	Tabs separate column values, and each line is a separate record. Column values can vary in length.
Character-separated values	Any printable character except single and double quotes can separate column values, and each line is a separate record. Column values can vary in length.
Fixed	No character separates column values. Instead, values start at the same position and have the same length in each line. The values appear in fixed columns if you display the file. Each line is a separate record.
Stream	No character separates column values nor records. The table is one long stream of bytes.

Comma-, tab-, and character-separated files are called character-delimited files because values are separated by a special character.

Configuring Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 “Quick Start Connect” on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See “Using a Connection String” on page 965 and Table 20-2 on page 967 for an alphabetical list of driver connection string attributes and their initial default values.

Data Source Configuration in the UNIX odbc.ini File

On UNIX and Linux, you must set up the proper ODBC environment before configuring data sources. See “Environment Configuration” on page 45 for basic setup information and “Environment Variables” on page 131 for more detail about this procedure.

Data sources for UNIX and Linux are stored in the system information file (by default, odbc.ini). If you have a Motif GUI environment on UNIX or Linux, you can configure and modify data sources through the DataDirect ODBC Data Source Administrator for UNIX/Linux (the UNIX ODBC Administrator) using a driver Setup dialog box. (See “Configuration Through the Administrator” on page 138 for a detailed explanation of the Administrator.)

If you do not have a GUI environment, you can configure and modify data sources directly by editing the odbc.ini file and storing default connection values there. See "Configuration Through the odbc.ini File" on page 141 for detailed information about the specific steps necessary to configure a data source.

Table 20-2 on page 967 lists driver connection string attributes that must be used in the odbc.ini file to set the value of connection options. Note that only the long name of the attribute can be used in the file. The default listed in the table is the initial default value when the driver is installed.

Data Source Configuration through a GUI

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box, as described in this section.

On UNIX and Linux, data sources are stored in the odbc.ini file. You can configure and modify data sources through the UNIX ODBC Administrator using a driver Setup dialog box, as described in this section.

NOTE: This book shows dialog box images that are specific to Windows. If you are using the drivers in the UNIX/Linux environments, the dialog box that you see may differ slightly from the Windows version. Windows-only and UNIX-only connection options are specifically noted by icons in the Setup dialog box descriptions.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed

by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure a Text data source:

1 Start the ODBC Administrator:

- On Windows, start the ODBC Administrator by selecting its icon from the DataDirect Connect program group.

- On UNIX and Linux, change to the *install_dir*/tools directory and, at a command prompt, enter:

  ```
  odbcadmin
  ```

  where *install_dir* is the path to the product installation directory.

2 Select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new user data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

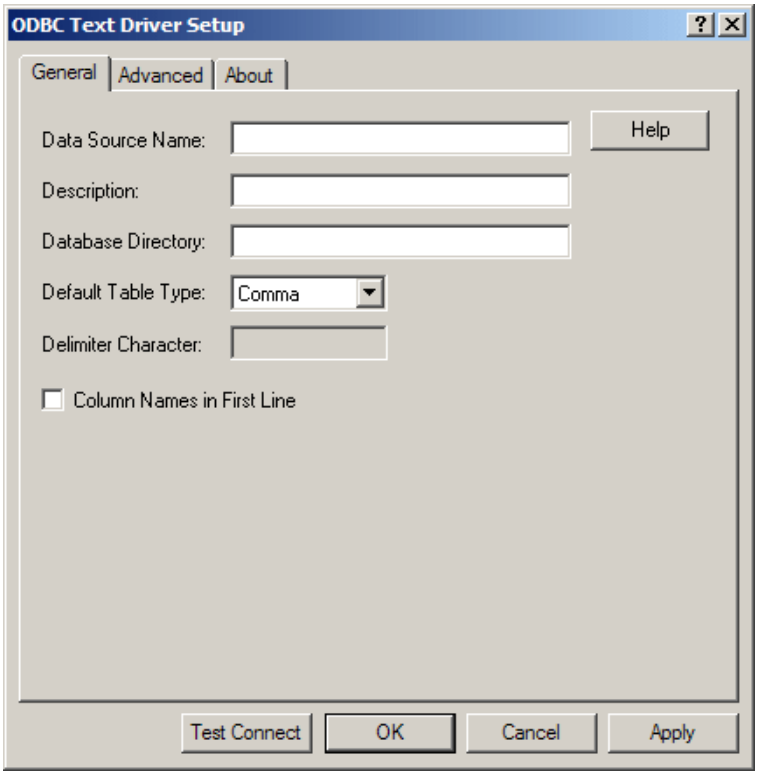
- **System DSN**: If you are configuring an existing system data source, select the data source name and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new system data source, click **Add** to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source file and click **Configure** to display the driver Setup dialog box.

If you are configuring a new file data source, click **Add** to display a list of installed drivers; then, select a driver. Click **Advanced** if you want to specify attributes; otherwise, click **Next** to proceed. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.

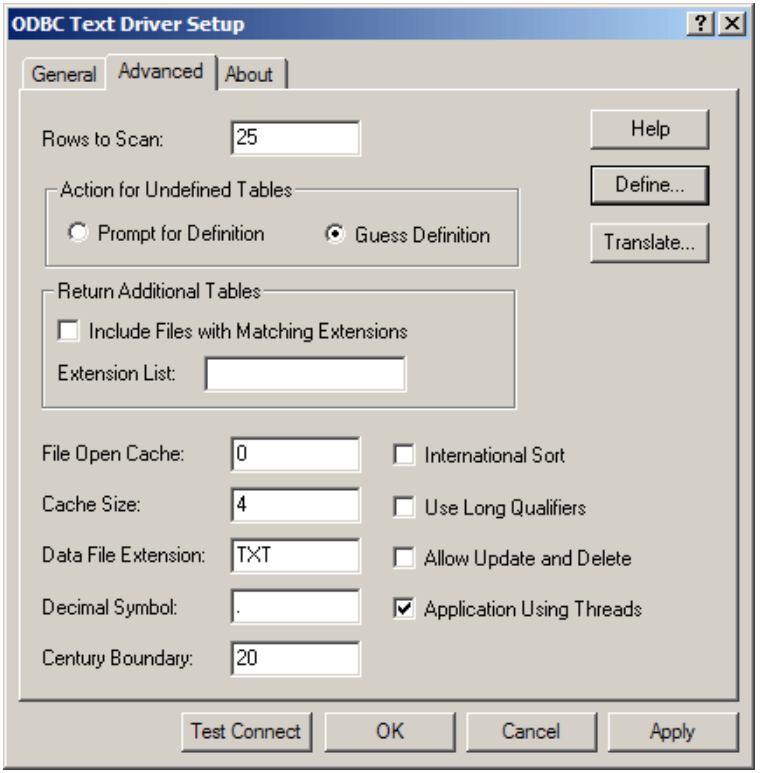


NOTE: The General tab displays the only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

3 On this tab, provide values for the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: General	Default
Data Source Name (see page 972)	None
Description (see page 975)	None
Database Directory (see page 973)	None
Default Table Type (see page 974)	Comma
Delimiter Character (see page 974)	None
Column Names in First Line (see page 971)	Disabled

4 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide values for any of the options in the following table; then, click **Apply.** The table provides links to descriptions of the connection options.

Connection Options: Advanced	Default
Rows to Scan (see page 978)	25
Action for Undefined Tables (see page 968)	GUESS
Include Files with Matching Extensions (see page 977)	Disabled
Extension List (see page 975)	None
File Open Cache (see page 976)	0
International Sort (see page 978)	Disabled
Cache Size (see page 970)	4
Use Long Qualifiers (see page 979) WINDOWS ONLY	Disabled
Data File Extension (see page 972)	TXT
Allow Update and Delete (see page 969)	Disabled
Decimal Symbol (see page 973)	. (Period)
Application Using Threads (see page 969)	Enabled
Century Boundary (see page 970)	20

Define: Click **Define** to define the structure of your text files as described in "Defining Table Structure on Windows" on page 980.

Translate: Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the Registry. DataDirect Technologies provides a translator named OEM to ANSI that translates your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box.

5 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box.

   - If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

6 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 20-2 lists the long and short names for each attribute, as well as the initial default value when the driver is first installed. You can specify either long or short names in the connection string.

An example of a DSN connection string with overriding attribute values for Text is:

```
DSN=Text1;DB=C:\TEXTDATA;TT=Comma
```

A FILEDSN connection string is similar except for the initial keyword:

```
FILEDSN=Text1.dsn;DB=C:\TEXTDATA;TT=Comma
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 TextFile (*.*);
DB=C:\TEXTDATA;TT=Comma
```

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name, along with its short name, is listed immediately underneath the GUI name. For example:

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

Table 20-2 lists the connection string attributes supported by the Text driver.

Table 20-2. Text Attribute Names

Attribute (Short Name)	Default
AllowUpdateAndDelete (AUD)	0 (Disabled)
ApplicationUsingThreads (AUT)	1 (Enabled)
CacheSize (CSZ)	4
CenturyBoundary (CB)	20
Database (DB)	None
DataFileExtension (DFE)	TXT
DataSourceName (DSN)	None
DecimalSymbol (CS)	. (Period)
Delimiter (DC)	, (Comma)

Table 20-2. Text Attribute Names (cont.)

Attribute (Short Name)	Default
Description (n/a)	None
ExtraExtensions (EE)	None
FileOpenCache (FOC)	0
FirstLineNames (FLN)	0 (Disabled)
IANAAppCodePage (IACP) UNIX ONLY	4 (ISO 8559-1 Latin-1)
IntlSort (IS)	0 (Disabled)
ScanRows (SR)	25
TableType (TT)	Comma
UndefinedTable (UT)	GUESS
UseLongQualifiers (ULQ) WINDOWS ONLY	0 (Disabled)

Action for Undefined Tables

Attribute: UndefinedTable (UT)

Description: Determines whether the driver prompts the user when it encounters a table for which it has no structure information.

Valid Values: PROMPT | GUESS

Specify PROMPT to prompt the user.

Specify GUESS for the driver to guess the format of the file.

Default: GUESS

GUI Tab: Advanced tab on page 963

Allow Update and Delete

Attribute AllowUpdateAndDelete (AUD)

Description Allows Update and Delete statements. Because Update and Delete statements cause immediate changes to a text file, only one connection at a time can operate on a file. Each update and delete on a text file can cause significant changes to the file, and performance may be degraded. Consider a more appropriate database form if performance is a significant factor.

Valid Values 0 | 1

If set to 1 (Enabled), text files are opened exclusively by the current connection.

If set to 0 (Disabled), Update and Delete statements are not allowed.

Default 0 (Disabled)

GUI Tab Advanced tab on page 963

Application Using Threads

Attribute ApplicationUsingThreads (AUT)

Description Determines whether the driver works with applications using multiple ODBC threads.

Valid Values 0 | 1

If set to 1 (Enabled), the driver works with single-threaded and multi-threaded applications.

If set to 0 (Disabled), the driver does not work with multi-threaded applications. If using the driver with

single-threaded applications, this value avoids additional processing required for ODBC thread-safety standards.

Default 1 (Enabled)

GUI Tab Advanced tab on page 963

Cache Size

Attribute CacheSize (CSZ)

Description The number of 64 KB blocks the driver uses to cache database records. The larger the number of blocks, the better the performance.

Valid Values 0 | *x*

where *x* is a positive integer that specifies the number of 64 KB blocks for caching.

If set to 0, no records are cached.

If set to *x*, the specified number of 64 KB blocks are set aside for caching. The maximum number of blocks you can set depends on the system memory available. If the cache size is greater than 0, when browsing backwards, you are not able to see updates made by other users until you run the Select statement again.

Default 4

GUI Tab Advanced tab on page 963

Century Boundary

Attribute CenturyBoundary (CB)

Description The cutoff year for century inference when converting two-digit dates to four-digit dates.

Valid Values *xx*

where *xx* is a two-digit number.

Two-digit dates that are less than the specified year number are converted to 20*xx*. Two-digit dates greater than or equal to the number are converted to 19*xx*. For example, using the default value of 20, a date of 19 will be interpreted as 2019 and a date of 21 is interpreted as 1921.

Default 20

GUI Tab Advanced tab on page 963

Column Names in First Line

Attribute FirstLineNames (FLN)

Description Determines whether the driver looks for column names in the first line of the file.

NOTE: The Column Names in First Line setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Valid Values 0 | 1

If set to 1 (Enabled), the driver looks for column names in the first line of the file.

If set to 0 (Disabled), the driver does not look for column names in the first line of the file.

Default 0 (Disabled)

GUI Tab General tab on page 963

Data File Extension

Attribute: DataFileExtension (DFE)

Description: A one- to three-character file name extension to use for data files.

Valid Values: *ext*

where *ext* is the name of the one- to three-character file name extension.

This value is used for all Create Table statements. Sending a Create Table using an extension other than the value specified for this option causes an error.

In other SQL statements, such as Select or Insert, users can specify an extension other than the one specified for this connection option. The Data File Extension value is used when no extension is specified.

Default: TXT

GUI Tab: Advanced tab on page 963

Data Source Name

Attribute: DataSourceName (DSN)

Description: 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 on page 963

Database Directory

Attribute Database (DB)

Description The directory that contains the data files.

Valid Values *database_directory*

where *database_directory* is the full path name of the directory in which the data files are stored. If no directory is specified, the current working directory is used.

Default None

GUI Tab General tab on page 963

Decimal Symbol

Attribute DecimalSymbol (CS)

Description The decimal separator used when data is stored.

Valid Values , | .

If set to Comma (,), the driver uses a comma as the decimal separator.

If set to Period (.), the driver uses a period as the decimal separator.

The international decimal symbol (.) must be used in DML statements and parameter buffers.

Default . (Period)

GUI Tab Advanced tab on page 963

Default Table Type

Attribute: TableType (TT)

Description: The type of text file (table) that is used when creating a new table and opening an undefined table.

NOTE: The Default Table Type setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Valid Values: Comma | Tab | Character | Fixed | Stream

The value chosen determines the type of text used for a table: comma-separated, tab-separated, character-separated, fixed length, or stream.

Default: Comma

GUI Tab: General tab on page 963

Delimiter Character

Attribute: Delimiter (DC)

Description: The character used as a delimiter for character-separated files.

NOTE: The Delimiter Character setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Valid Values: *x*

where *x* is any printable character except single quotes, double quotes, or semicolons.

Note that it is possible to specify a semicolon if you configure the data source using the Windows ODBC Administrator.

Default: , (Comma)

GUI Tab: General tab on page 963

Description

Attribute Description (n/a)

Description 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 on page 963

Extension List

Attribute ExtraExtensions (EE)

Description A comma-separated list of file name extensions for the files that you want returned in addition to the extension specified in the Data File Extension field.

NOTE: You must have also enabled the Files with Matching Extensions option.

Valid Values *ext* | NONE

where *ext* is a file name extension.

To have files with no extensions returned, specify NONE. For example, if some of your files have the extensions TXT and CSV and others have no extension, specify `TXT,CSV,NONE`.

By default, when an application requests a list of tables, only files that have been defined are returned.

Default None

GUI Tab Advanced tab on page 963

File Open Cache

Attribute FileOpenCache (FOC)

Description The maximum number of used file handles to cache.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, no file open caching is performed.

If set to *x*, when a user opens and closes *x* tables, the tables are not actually closed. The driver keeps them open so that if another query uses one of these tables, the driver does not have to perform another open, which is expensive. The advantage of file open caching is improved performance. The disadvantage is that a user who tries to open the file exclusively may get a file locking conflict even though no one appears to have the file open.

Default 0 (No File Open Caching)

GUI Tab Advanced tab on page 963

IANAAppCodePage

Attribute IANAAppCodePage (IACP)

Description An Internet Assigned Numbers Authority (IANA) value. You must specify a value for this option if your application is not Unicode-enabled and/or if your database character set is not Unicode (refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for details). The value you specify must match the database character encoding and the system locale.

The Driver Manager checks for the value of IANAAppCodePage in the following order:

- In the connection string
- In the Data Source section of the system information file (odbc.ini)
- In the ODBC section of the system information file (odbc.ini)

Valid Values *IANA_code_page*

where *IANA_code_page* is one of the valid values listed in Chapter 1 "Values for the Attribute IANAAppCodePage" in the *DataDirect Connect Series for ODBC Reference*. The value must match the database character encoding and the system locale.

Default 4 (ISO 8559-1 Latin-1)

GUI Tab Advanced tab on page 963

Include Files with Matching Extensions

Attribute n/a

Description Enables the driver to return files with a given file name extension in addition to the extension specified through the Data File Extension option. After enabling this option, specify the file name extensions through the Extension List option.

Valid Values 0 | 1

If set to 1 (Enabled), the driver returns files with the file name extensions specified through the Extension List and Data File Extension options.

If set to 0 (Disabled), the driver returns only files with the file name extension specified through the Data File Extension option.

Default 0 (Disabled)

GUI Tab Advanced tab on page 963

International Sort

Attribute IntlSort (IS)

Description Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values 0 | 1

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default 0 (Disabled)

GUI Tab Advanced tab on page 963

Rows to Scan

Attribute ScanRows (SR)

Description The number of rows in a text file that the driver scans to determine the data types in the file.

NOTE: The Rows to Scan setting applies only to tables not previously defined. It also determines the attributes of new tables created with the Create Table statement.

Valid Values 0 | *x*

where *x* is a positive integer.

If set to 0, all rows in the file are scanned.

If set to *x*, *x* rows are scanned to determine the data types in a file.

Default 25

GUI Tab Advanced tab on page 963

Use Long Qualifiers

Attribute UseLongQualifiers (ULQ)

Description Determines whether the driver uses long path names.

Valid Values 0 | 1

If set to 1 (Enabled), path names can be a maximum of 255 characters.

If set to 0 (Disabled), path names can be a maximum of 128 characters.

Default 0 (Disabled)

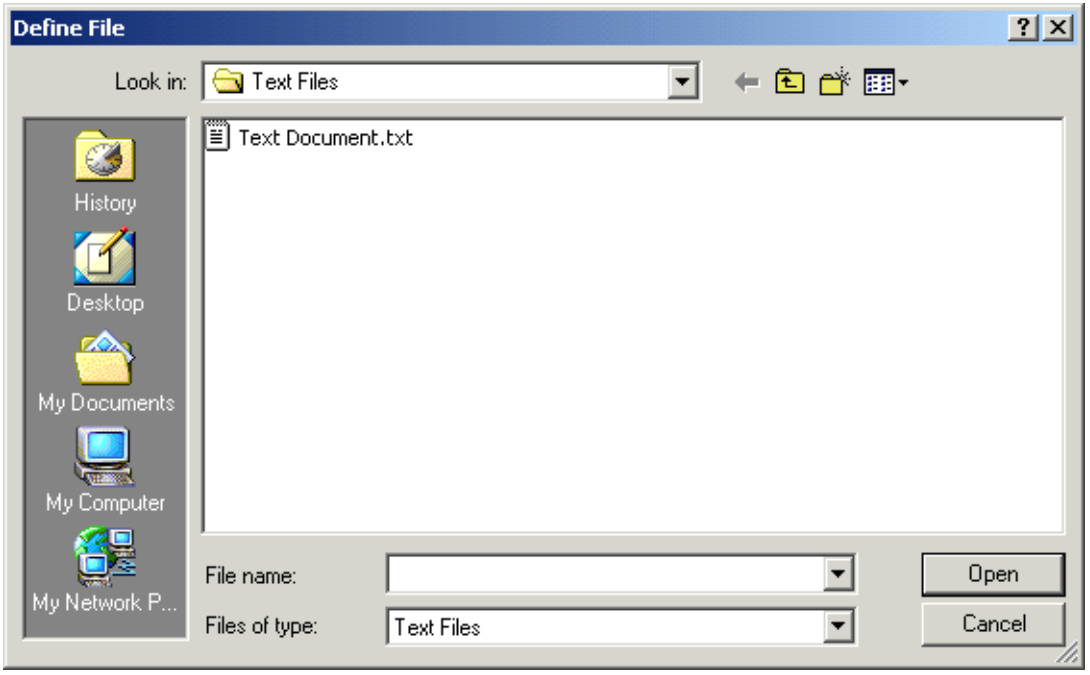
GUI Tab Advanced tab on page 963

Defining Table Structure on Windows

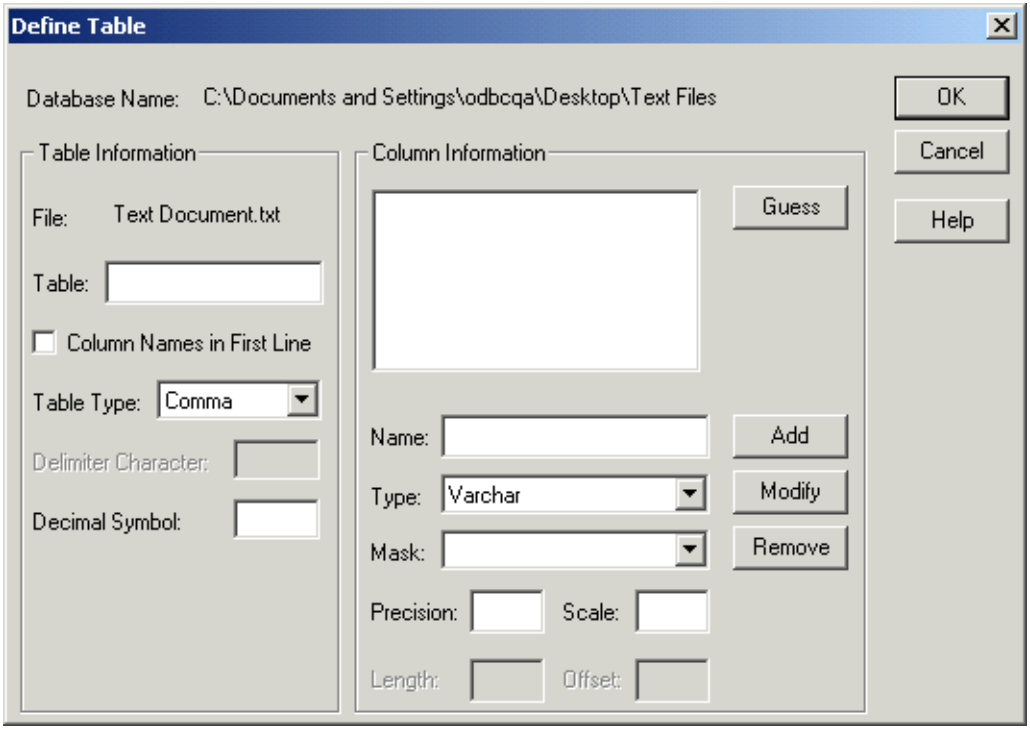
Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory (the driver can attempt to guess the names and types of the columns), this feature is extremely useful.

To define the structure of a file:

1 Display the ODBC Text Driver Setup dialog box through the ODBC Administrator. Click the **Advanced** tab; then, click **Define** to display the Define File dialog box.



2 Select the correct file and click **Open** to display the Define Table dialog box.



Database Name: This field displays the name of the database directory that you selected in the Define File dialog box.

File: This field displays the name of the file that you selected in the Define File dialog box.

Table: Type a table name in the Table field. This name specifies the table name associated with the text file you selected earlier. The name can be a maximum of 32 characters and must be unique. This name is returned by SQLTables. By default, it is the file name without its extension (for example, Trc_read).

Column Names in First Line: Select this check box if the first line of the file contains column names; otherwise, do not select this box.

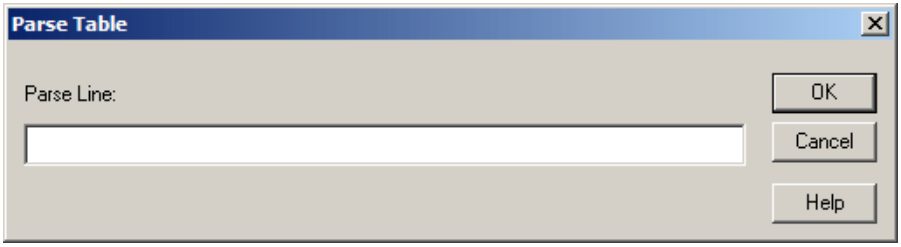
Table Type: Select the type of text file, either comma, tab, fixed, character, or stream.

Delimiter Character: If the table type is Character, type the delimiter used in character-separated files. The value can be any printable character except single and double quotes.

Decimal Symbol: Type the decimal separator used when data is stored. Valid values are a comma or a period. The international decimal symbol (.) must be used in DML statements and parameter buffers.

3 If you specified a comma-separated, tab-separated, or character-separated type in the Table Type field, the Guess/Parse button displays Guess. Click **Guess** to have the driver guess at the column names and display them in the list box of the Column Information pane.

If you specified a fixed-length or stream type in the Table Type field, the Guess/Parse button displays Parse. Click **Parse** to have the driver display the Parse Table dialog box and define the table columns.



This dialog box displays the first line of the file. You must mark where *each* field begins and ends by enclosing it in square brackets [ ]. These brackets indicate the position and length of each field value in the record. Click **OK** to close the Parse Table dialog box. The driver will suggest column names in the list box of the Column Information pane.

4 If you do not want the driver to guess or parse, enter values in the following fields to define each column. Click **Add** to add the column name to the Column Information box.

Name: Type the name of the column.

Type: Select the data type of the column. If the field type is Date, the Mask field is enabled and you must select a date mask or type one in. See "Date Masks" on page 986 for more information.

Mask: Select a date mask. If you selected Date for the Type field, you must select a date mask for the field or type one in. See "Date Masks" on page 986 for more information.

Precision: Type the precision of the column. The precision of numeric data types is defined as the maximum number of digits used by the data type of the column. For character types, this is the length in characters of the data. Note that the precision and scale values determine how numeric data is to be returned.

Scale: Type the scale of the column. The scale of numeric data types is defined as the maximum number of digits to the right of the decimal point. Note that the precision and scale values determine how numeric data is to be returned.

Length: If you specified a fixed-length table type, type the length, which is the number of bytes the data takes up in storage.

Offset: If you specified a fixed-length table type, type the offset, which is the number of bytes from the start of the table to the start of the field.

5 To modify an existing column definition, select the column name in the Column Information box. Modify the values for that column name; then, click **Modify**.

6 To delete an existing column definition, select a column name in the Column Information box and click **Remove**.

7 Click **OK** to define the table.

Defining Table Structure on UNIX and Linux

Because text files do not all have the same structure, the driver provides the option to define the structure of an existing file. Although defining the structure is not mandatory, because the driver can attempt to guess the names and types of the columns, this feature is extremely useful.

To define the structure of a text file, you create a QETXT.INI file using any plain text editor, such as vi. The file name must be in uppercase. All of the tables you want to define are specified in the QETXT.INI file. When you specify table attributes in QETXT.INI, you override the attributes specified in the system information file (odbc.ini) or in the connection string.

To define the QETXT.INI file:

1 Create a [Defined Tables] section and list all of the tables you are defining. Specify the text file name (in either upper or lowercase, depending on the file) followed by the name you want to give the table, for example:

```
emptext.txt=EMP
```

Table names can be up to 32 characters in length and cannot be the same as another defined table in the database. This name is returned by SQLTables. By default, it is the file name without its extension.

2 For each table listed in the [Defined Tables] section, you must specify the text file (FILE=), the table type (TT=), whether the first line of the file contains column names (FLN=), and the delimiter character (DC=).

- Specify the text file name. For example:

  ```
  FILE=emptext.txt
  ```

- To define the table type, specify how the fields are separated (comma, tab, fixed, or character). For example:

  ```
  TT=COMMA
  ```

- If the table type is CHARACTER, specify the delimiter character. The value can be any printable character except single and double quotes. For example, if the fields are separated by comma:

  ```
  DC=,
  ```

- Specify whether the first line of the file contains column names, using 1 for yes and 0 for no. For example:

  ```
  FLN=0
  ```

3 Define the fields in the table, beginning with FIELD1. For each field, specify the field name, field type, precision, scale, length, offset (for fixed tables), and date/time mask. See "Date Masks" on page 986 for information about masks.

Separate the values with commas. For example, to define two fields:

```
FIELD1=EMP_ID,VARCHAR,6,0,6,0,
FIELD2=HIRE_DATE,DATE,10,0,10,0,m/d/yy
```

4 Save the file as QETXT.INI. The driver looks for this file in the directory specified by the Database attribute in odbc.ini, or in the current directory.

Example of QETXT.INI

The following is an example of a QETXT.INI file. This file defines the structure of the emptext.txt file, which is a sample data file shipped with the DataDirect ODBC Text file.

```
[Defined Tables]
emptext.txt=EMP

[EMP]
FILE=emptext.txt
FLN=1
TT=Comma
FIELD1=FIRST_NAME,VARCHAR,10,0,10,0,
FIELD2=LAST_NAME,VARCHAR,9,0,9,0,
FIELD3=EMP_ID,VARCHAR,6,0,6,0,
FIELD4=HIRE_DATE,DATE,10,0,10,0,m/d/yy
FIELD5=SALARY,NUMERIC,8,2,8,0,
FIELD6=DEPT,VARCHAR,4,0,4,0,
FIELD7=EXEMPT,VARCHAR,6,0,6,0,
FIELD8=INTERESTS,VARCHAR,136,0,136,0,
```

Date Masks

Date masks tell the driver how a date is stored in a text file. When a value is inserted into a text file, the date is formatted so that it matches the mask. When reading a text file, the driver converts the formatted date into a date data type.

Table 20-3 lists the symbols to use when specifying the date mask.

Table 20-3. Date Masks for Text Driver

Symbol	Description
m	Output the month's number (1–12).
mm	Output a leading zero if the month number is less than 10.
mmm, Mmm, MMM	Output the three-letter abbreviation for the month depending on the case of the Ms (for example, jan, Jan, JAN).
mmmm, Mmmm, MMMM	Output the name of the full month depending on the case of the Ms (for example, january, January, JANUARY).
d	Output the day number (1–31).
dd	Output a leading zero if the day number is less than 10.
ddd, Ddd, DDD	Output the three-letter day abbreviation depending on the case of the Ds (for example, mon, Mon, MON).
dddd, Dddd, DDDD	Output the name of the full day depending on the case of the Ds (for example, monday, Monday, MONDAY).
yy	Output the last two digits of the year.
yyyy	Output the full four digits of the year.
J	Output the Julian value for the date. The Julian value is the number of days since 4712 BC.
\ - . : , (space)	Special characters used to separate the parts of a date.
\	Output the next character. For example, if the mask is mm/dd/yyyy \A\D, the value appears as 10/01/2003 AD in the text file.
"string", 'string'	Output the string in the text file.

Table 20-4 shows some example date values, masks, and how the date appears in the text file.

Table 20-4. Date Mask Examples

Date	Mask	Value
2003-10-01	yyyy-mm-dd	2003-10-01
	m/d/yy	10/1/03
	Ddd, Mmm dd, yyyy	Fri, Oct 01, 2003

Data Types

Table 20-5 shows how the text file data types are mapped to the standard ODBC data types.

Table 20-5. Text Data Types

Text	ODBC
Numeric	SQL_NUMERIC
Date	SQL_TYPE_DATE
Varchar	SQL_VARCHAR

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Select Statement

You use a SQL Select statement to specify the columns and records to be read. All of the Select statement clauses described in Chapter 10 "SQL for Flat-File Drivers" in the *DataDirect Connect Series for ODBC Reference* are supported by the Text driver.

Alter Table Statement

The Text driver supports the Alter Table statement to add one or more columns to a table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {ADD column_name data_type |
ADD(column_name data_type [, column_name data_type]... ) |
DROP[COLUMN] column_name}
```

table_name is the name of the table to which you are adding or dropping columns.

column_name assigns a name to the column you are adding or specifies the column you are dropping.

data_type specifies the native data type of each column you add.

For example, to add two columns to the emp table:

```
ALTER TABLE emp (ADD startdate date, dept varchar(10))
```

You cannot add columns and drop columns in a single statement, and you can drop only one column at a time. For example, to drop a column:

```
ALTER TABLE emp DROP startdate
```

The Alter Table statement fails when you attempt to drop a column upon which other objects, such as indexes or views, are dependent.

ODBC Conformance Level

The Text driver supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver also supports the minimum SQL grammar.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the Text driver. In addition, the following function is supported: SQLSetPos.

Number of Connections and Statements Supported

Text files support multiple connections and multiple statements per connection.

21 The XML Driver

The DataDirect Connect *for* ODBC XML driver (the XML driver) supports:

Tabular- and hierarchical-formatted XML documents that can be accessed from either a local file system, a web server, or a web service. The three main types of tabular-formatted files that the driver supports are Microsoft Data Islands, ADO 2.5 persisted files, and DataDirect Format.

See "Supported Tabular Formats for XML Documents" on page 992 for more details.

The XML driver is 32-bit only and is supported in the Windows environments. See "Environment-Specific Information" on page 59 for detailed information about the environments supported by this driver.

The XML driver includes a SQL Engine that provides ANSI SQL-92 support. The following table lists the SQL statements that the driver supports for the different types of file formats.

File Format	Select	Create/Drop	Insert	Update	Delete
Tabular, Microsoft Data Islands	X	X	X	X	X
Tabular, ADO 2.5 Persisted	X	X	X	X	X
Tabular, DataDirect	X	X	X	X	X
Tabular, other formats	X		X	X	X
Hierarchical	X				

See "SQL Supported" on page 1036 for more information.

Refer to the readme file shipped with your DataDirect Connect product for the file name of the XML driver.

Driver Requirements

You must have Internet Explorer 5 or higher installed. You must also have the Microsoft XML parser, msxml4.dll, not a higher version, installed. If you need to download the file, go to the site:

http://www.microsoft.com

On the Microsoft site, search on "msxml4.dll". Select the link for downloading the parser.

Supported Tabular Formats for XML Documents

The three main XML tabular-formats that the XML driver can access are described in Table 21-1. In some instances, you may need to define hints to help the XML driver read the tabular-format of an XML document correctly. See "Configure Location Dialog Box Descriptions" on page 1019.

Table 21-1. Common Tabular Formats for XML Documents

Format	Description

ADO 2.5 persisted files	These files are identified by a unique schema namespace URL. Although ADO uses the same data types defined by XML-Data, the data types use extensions, such as adding a maximum column width for string columns. ADO 2.5 persisted files are identified by the following unique XML element:
	`<xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882"`
	`xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"`
	`xmlns:rs="urn:schemas-microsoft-com:rowset"`
	`xmlns:z="#RowsetSchema">`

Table 21-1. Common Tabular Formats for XML Documents *(cont.)*

Format	Description
DataDirect Format	This XML format conforms to the W3C recommendation for XML schema, Working Draft April 07, 2000. These files are identified by the following unique XML element (schema namespace URL): `<table targetNamespace= "http://www.merant.com/namespaces/datadirect/xmlrecordset" xsi:schemaLocation= "<http://www.merant.com/namespaces/datadirect/xmlrecordset/EMP.xml>" xmlns="http://www.w3.org/1999/XMLSchema" xmlns:xsi= "http://www.w3.org/1999/XMLSchema-instance" xmlns:rs= "http://www.merant.com/namespaces/datadirect/xmlrecordset">`
Microsoft Data Islands	These islands are identified by the <XML> tag in an HTML document. The Data Island can be embedded in the HTML document. Data Islands can include the following Schema definition and namespace: `<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes">`

Hierarchical-Formatted XML Document Support

The XML driver can be configured so that it supports hierarchical-formatted documents. In this case, the driver assumes that the document that it is accessing can contain more than one table. The driver scans the document to locate all tables; the available tables are visible through a SQLTables operation. Then, the driver does a second scan to gather each table's column information and to determine a data type for each column.

Let's look at an example of a hierarchical document and discuss the results.

```
<?xml version="1.0"?>
  <purchaseOrder orderDate="1999-10-20">
    <shipTo country="US">
      <name>Alice Smith</name>
      <street>123 Maple Street</street>
      <city>Mill Valley</city>
      <state>CA</state>
      <zip>90952</zip>
    </shipTo>
    <billTo country="US">
       <name>Robert Smith</name>
       <street>8 Oak Avenue</street>
       <city>Old Town</city>
       <state>PA</state>
       <zip>95819</zip>
    </billTo>
    <comment>Hurry, my lawn is going wild!</comment>
    <items>
       <item partNum="872-AA">
          <productName>Lawnmower</productName>
          <quantity>1</quantity>
          <USPrice>148.95</USPrice>
          <comment>Confirm this is electric</comment>
       </item>
       <item partNum="926-AA">
          <productName>Baby Monitor</productName>
          <quantity>1</quantity>
          <USPrice>39.98</USPrice>
          <shipDate>1999-05-21</shipDate>
       </item>
    </items>
  </purchaseOrder>
```

First, the XML driver returns two tables: "purchaseOrder" and "items." Two tables are returned because two items are found for a single purchase order. The XML driver found commonality of child elements.

Second, the XML driver determines which columns are in a specific table. An _ID column, which is essentially a primary key, is automatically generated for each table. If a table is determined to be a child of another table, then it is given a second generated column. The name of this column is prefixed with the parent table's name and ends with _ID, for example, _purchaseOrder_ID.

Consider the previous example document. The items table will receive two generated columns, _ID and _purchaseOrder_ID, which are assigned an integer data type. The purchaseOrder table receives only the _ID column, because it does not have a parent table.

The tables returned from the example file include the following columns:

Table	Columns	
items	_ID	quantity
	_purchaseOrder_ID	USPrice
	partNum	comment
	productName	shipDate
purchaseOrder	_ID	billTo_country
	orderDate	billTo_name
	shipTo_country	billTo_street
	shipTo_name	billTo_city
	shipTo_street	billTo_state
	shipTo_city	billTo_zip
	shipTo_state	comment
	shipTo_zip	

Column Data Types

The XML driver determines the column data types by inspecting the column values. The data type determination limits its data types to a subset of the DataDirect Format data types, as listed in the following table. For a complete list of DataDirect Format data types, see Table 21-6 on page 1034.

Data Type	Sample Values
wvchar	"Foo", "best320"
varbinary	"27AB2F9C"
int	"34","-7000"
unsignedint	"0", "123456789"
long	"-12345678012345"
unsignedlong	"123456789012345"
boolean	"true", "false"
date	1963-12-19
time	10:09:58
timeinstant	1963-12-19T10:09:58
decimal	1245.678

Defining Locations

When configuring an XML data source, you must define the location of the XML or HTML documents that the driver will access. The locations can be either from a local file system or from a Web server.

The types of locations are:

Folder	Implies that each XML file is a single table. When defining a Folder location, you specify only a directory as the location (not a directory and a file name), for example, C:\xmlsample.
XML Document	Implies that the full path to the XML document, including the XML file name, is the location. Using this type of location, each document can have one or more tables and can be a hierarchical-formatted XML document. When defining an XML Document location, you specify a path and an XML file name as the location, for example: C:\xmlsample\file.xml You can also specify a web service through a URL, for example: http://xxx.company.com/search=XML&mode=books
HTML Document	Implies the use of an HTML document with embedded XML Data Islands. Using this type of location, each document can have one or more tables. When defining an HTML Document location, you specify a path and an HTML file name, for example, C:\htmlsample\file.html, as the location.

Specifying Table Names in SQL Statements

When defining locations, you specify a name for the location along with a directory, or path and file name. For example, suppose you define two locations for a data source, a Folder location and an XML Document location. The Folder location is on a local filing system and the XML Document location is on a web server with a URL prefix of http://www.acme.com/xmldata.

For example:

The Folder location:
c:\xmldata\xmlsample as LOC1

The XML Document location:
http://www.acme.com/xmldata/doc.xml as LOC2

For complete information about how to configure locations in an XML data source, see "Configuring and Connecting to Data Sources" on page 1000.

If you are connected to this data source and the data source had the "Show Manufactured Schemas" option set as the Schema Mode (see the Schema Mode option under "Configuring and Connecting to Data Sources" on page 1000) and then you performed an unqualified SQLTables operation, you would get the following results.

Schema name	Table name
LOC1#	FILE1
LOC1#	FILE2
LOC2#	TABLE1
LOC2#	TABLE2

Location names are fabricated into the schema name by adding a # symbol to the end of the location name.

NOTE: If you had the "Show Virtual Schemas" option set, the above table would have "XML" listed in the Schema name column.

To fully qualify a table name in a SQL statement, you could use the following:

LOC1#.FILE1
or
XML.FILE1

LOC2#.TABLE2
or
XML.TABLE2

This design gives you a simpler table name qualifier. This is an important advantage given the complexity of URL names, and the requirement to double quote them in SQL statements. For example, the following query uses a fully qualified table name for an XML Document location:

```
SELECT * FROM "http://www.acme.com/xmldata/doc.xml#TABLE2"
WHERE productName='lawnmower'
```

Compare that to the same query using a location name:

```
SELECT * FROM LOC2#.TABLE2 WHERE productName='lawnmower'
```

Another example demonstrating the Folder location is as follows:

```
SELECT * FROM "c:\xmldata\xmlsample\FILE1.XML" WHERE
productName='lawnmower'
```

Compare that to the same query using a location name:

```
SELECT * FROM LOC1#.FILE1 WHERE productName='lawnmower'
```

Configuring and Connecting to Data Sources

After you install the driver, you configure data sources to connect to the database. See Chapter 1 "Quick Start Connect" on page 41 for an explanation of different types of data sources. The data source contains connection options that allow you to tune the driver for specific performance. If you want to use a data source but need to change some of its values, you can either modify the data source or override its values at connection time through a connection string.

If you choose to use a connection string, you must use specific connection string attributes. See "Using a Connection String" on page 1008 and Table 21-2 on page 1011 and Table 21-3 on page 1019 for an alphabetical list of driver connection string attributes and their initial default values.

On Windows, data sources are stored in the Windows Registry. You can configure and modify data sources through the ODBC Administrator using a driver Setup dialog box.

When the driver is first installed, the values of its connection options are set by default. These values appear on the driver Setup dialog box tabs when you create a new data source. You can change these default values by modifying the data source. In the following procedure, the description of each tab is followed by a table that lists the connection options for that tab and their initial default values. This table links you to a complete description of the options and their connection string attribute equivalents. The connection string attributes are used to override the default values of the data source if you want to change these values at connection time.

To configure an XML data source:

1 Start the ODBC Administrator by selecting its icon from the DataDirect Connect program group; then, select a tab:

- **User DSN**: If you are configuring an existing user data source, select the data source name on the User DSN tab and click **Configure** to display the driver Setup dialog box.

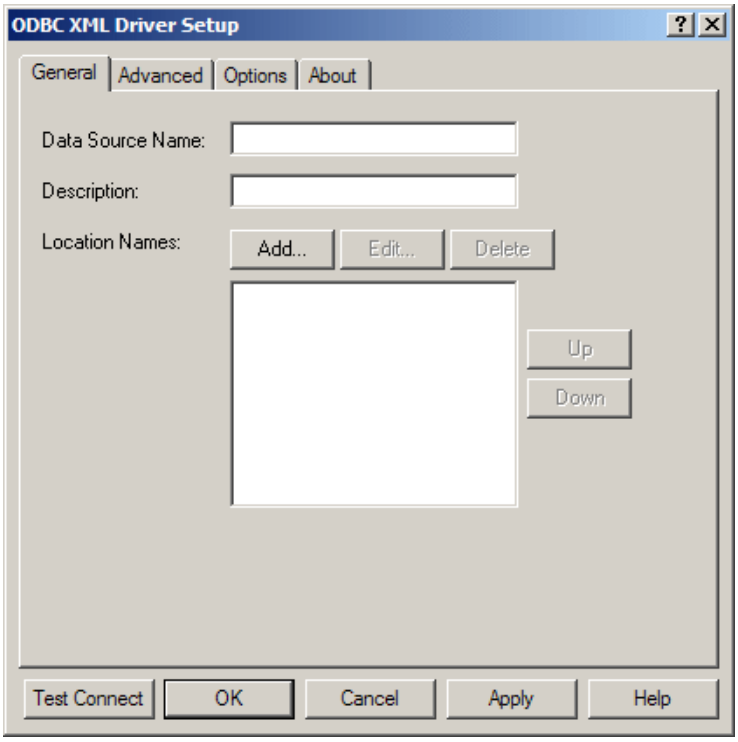
  If you are configuring a new user data source, click **Add** on the User DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **System DSN**: To configure a new system data source, click **Add** on the System DSN tab to display a list of installed drivers. Select the driver and click **Finish** to display the driver Setup dialog box.

- **File DSN**: If you are configuring an existing file data source, select the data source name on the File DSN tab and click **Configure** to display the driver Setup dialog box.

  If you are configuring a new file data source, click **Add** on the File DSN tab to display a list of installed drivers. Select the driver and click **Next**. Specify a name for the data source and click **Next**. Verify the data source information; then, click **Finish** to display the driver Setup dialog box.

The General tab of the Setup dialog box appears by default.



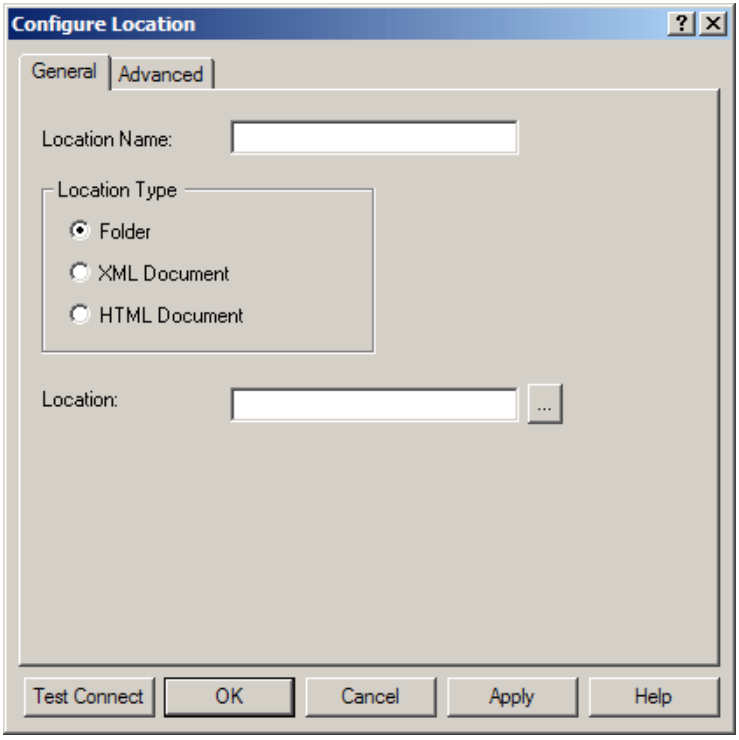
NOTE: The General tab displays only fields that are required for creating a data source. The fields on all other tabs are optional, unless noted otherwise.

2 On the General tab, provide the following information; then, click **Apply.**

Connection Options: General	Default
Data Source Name (see page 1012)	None
Description (see page 1012)	None
Location Names (see page 1013)	None

3 If you want to edit or delete a location name, or change its position in the list, select it; then, click **Edit**, **Delete**, **Up**, or **Down** as appropriate.

4 If you want to define a location, click **Add**. The Configure Location dialog box appears.



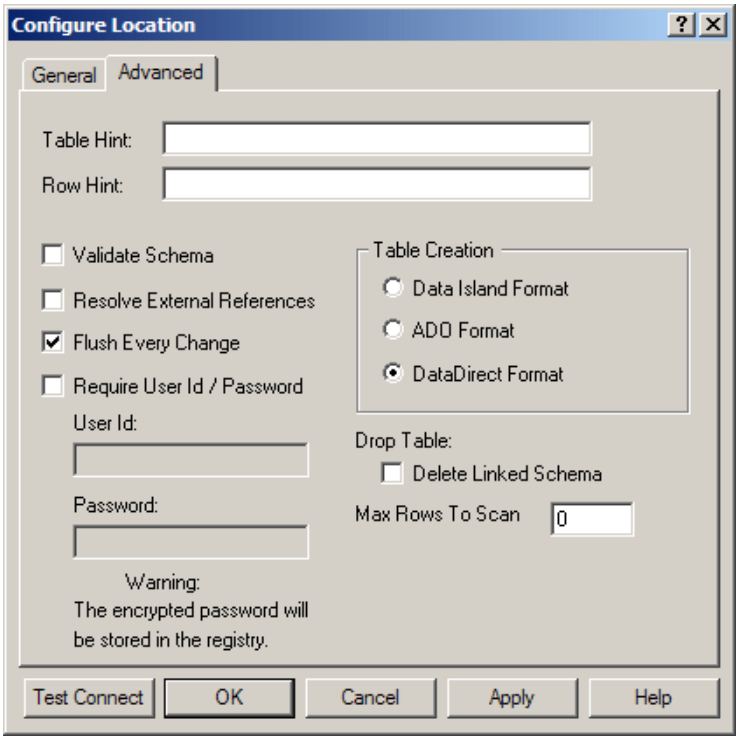
5 On the General tab of the Configure Location dialog box, provide the following required information; then, click **Apply**.

Configure Location Options: General	Default
Location Name (see page 1022)	None
Location Type (see page 1022)	None
Location (see page 1021)	localhost

Location: Either type the full path to the location you are defining or click the select button:

to select a path.

6 Optionally, click the **Advanced** tab of the Configure Location dialog box to specify additional information about the location you are defining.



On this tab, provide any of the following optional information; then, click **Apply**.

Configure Location Options: Advanced	**Default**
Table Hint (see page 1026)	None
Row Hint (see page 1025)	None
Validate Schema (see page 1027)	Disabled
Resolve External References (see page 1025)	Disabled

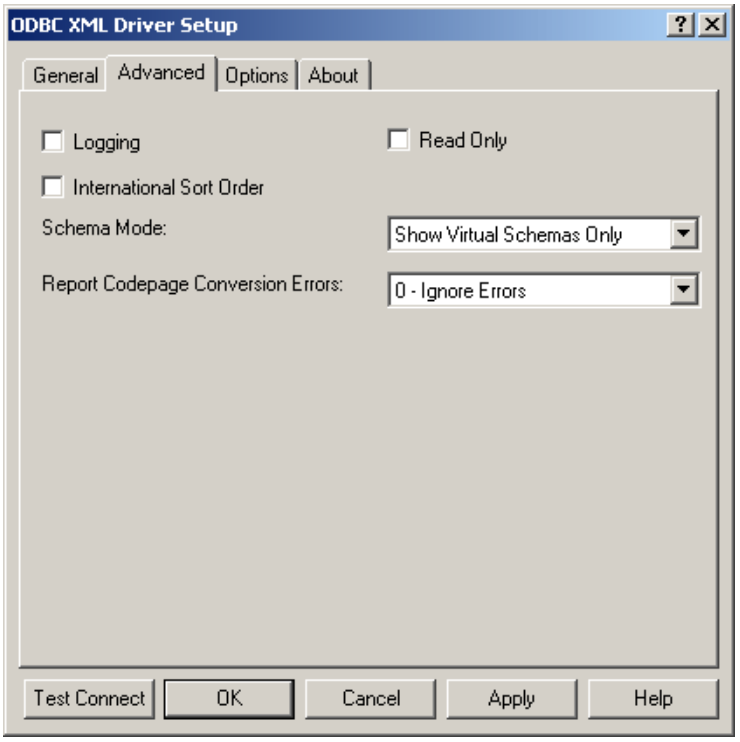
Configure Location Options: Advanced *(cont.)*	Default
Flush Every Change (see page 1020)	Enabled
Require User ID/Password (see page 1024)	Disabled
User ID (see page 1027)	None
Password (see page 1023)	None
Table Creation (see page 1026)	DataDirect Format
Delete Linked Schema (see page 1020)	Disabled
Max Rows to Scan (see page 1023)	0

7 You can click **Test Connect** to attempt to connect to the location.

- If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
- If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

8 Click **OK** to return to the ODBC XML driver Setup dialog box or **Cancel**. If you click **OK**, the values you have specified become the defaults for this location.

9 Optionally, click the **Advanced** tab of the ODBC XML driver Setup dialog box to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

Connection Options: Advanced	Default
Logging (see page 1014)	Disabled
International Sort Order (see page 1013)	Disabled
Schema Mode (see page 1016)	Show Virtual Schemas Only
Report Codepage Conversion Errors (see page 1015)	0 - Ignore Errors
Read Only (see page 1014)	Disabled

10 Optionally, click the **Options** tab to specify data source connection values.

Driver Options: Type configuration options specific to the XML driver.

Connection Options: Options	Default
Driver Options (see page 1012)	Disabled

WARNING: The properties you set in the Options tab override other properties for this session only and can adversely affect the operation of the XML driver. Use only authorized entries. For information about authorized entries for the Options tab, contact DataDirect Technologies technical support.

11 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection options specified in the driver Setup dialog box. A logon dialog box appears (see "Using a Logon Dialog Box" on page 1010 for details). Note that the information you enter in the logon dialog box during a test connect is not saved.

   - If the driver can connect, it releases the connection and displays a `connection established` message. Click **OK**.
   - If the driver cannot connect because of an incorrect environment or connection value, it displays an appropriate error message. Click **OK**.

12 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

Using a Connection String

If you want to use a connection string for connecting to a database, or if your application requires it, you must specify either a DSN (data source name), a File DSN, or a DSN-less connection in the string. The difference is whether you use the DSN=, FILEDSN=, or the DRIVER= keyword in the connection string, as described in the ODBC specification. A DSN or FILEDSN connection string tells the driver where to find the default connection information. Optionally, you may specify *attribute*=*value* pairs in the connection string to override the default values stored in the data source.

The DSN connection string has the form:

```
DSN=data_source_name[;attribute=value[;attribute=value]...]
```

The FILEDSN connection string has the form:

```
FILEDSN=filename.dsn[;attribute=value[;attribute=value]...]
```

The DSN-less connection string specifies a driver instead of a data source. All connection information must be entered in the connection string because no data source is storing the information.

The DSN-less connection string has the form:

```
DRIVER=[{]driver_name[}][;attribute=value[;attribute=
value]...]
```

Table 21-2 and Table 21-3 give the names and descriptions of the attributes, as well as the initial default value when the driver is first installed.

An example of a DSN connection string with overriding attribute values for XML is:

```
DSN=XML FILES;LOC1.Create Type=ADO25;Logging=1
```

A FILEDSN connection string is similar except for the initial keyword:

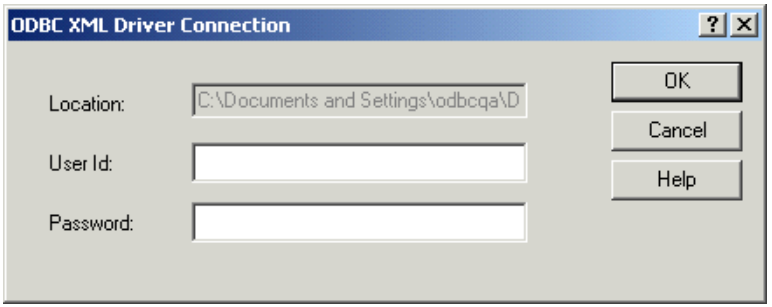
```
FILEDSN=XML.dsn;LOC1.Create Type=ADO25;Logging=1
```

A DSN-less connection string must provide all necessary connection information:

```
DRIVER=DataDirect 6.0 XML;
LOC1={DataDirect Closed XML ADO Provider}
```

Using a Logon Dialog Box

Some ODBC applications display a Logon dialog box when you are connecting to a data source. For XML, the dialog box is as follows:



This dialog box appears for each password-protected location that you have defined for the data source.

In this dialog box, provide the following information:

1. Type your user ID and password in the appropriate fields for the Location that appears in the Location field.
2. Click **OK** to connect to the data source.

Connection Option Descriptions

The following connection option descriptions are listed alphabetically by the GUI name that appears on the driver Setup dialog. The connection string attribute name is listed immediately underneath the GUI name. For example:

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors

In most cases, the GUI name and the attribute name are the same; however, some exceptions exist. If you need to look up an option by its connection string attribute name, please refer to the alphabetical table of connection string attribute names.

Also, a few connection string attributes, for example, Password, do not have equivalent options that appear on the GUI. They are in the list of descriptions alphabetically by their attribute names.

NOTE: XML driver connection string attributes do not use short name equivalents.

Driver Setup Dialog Box Descriptions

Table 21-2 lists the connection string attributes associated with General, Advanced, and Options tabs of the XML driver Setup dialog box. The descriptions themselves are listed below the table.

Table 21-2. XML Attribute Names

Attribute	Default
DataSourceName	None
Description	None
International Sort	Disabled
Location	None
Logging	Disabled
ReadOnly	Disabled
ReportCodepageConversionErrors	0 (Ignore Errors)
Show Manufactured Schemas	0 (Disabled)
Show Virtual Schemas	1 (Enabled)

Data Source Name

Attribute DataSourceName

Description 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 on page 1002

Description

Attribute Description

Description 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 on page 1002

Driver Options

Attribute n/a

Description Type configuration options specific to the XML driver.

Valid Values WARNING: The properties you set in the Options tab override other properties for this session only and can adversely affect the operation of the XML driver. Use only authorized entries. For information about authorized entries for the Options tab, contact DataDirect Technologies technical support.

Default None

GUI Tab Options tab on page 1007

International Sort Order

Attribute International Sort

Description Uses international sort order as defined by your operating system when you issue a Select statement with an Order By clause.

Valid Values 0 | 1

If set to 1 (Enabled), this order is always alphabetic, regardless of case; the letters are sorted as "A, b, C." Refer to your operating system documentation concerning the sorting of accented characters.

If set to 0 (Disabled), ASCII sort order is used. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" is sorted as "A, C, b."

Default 0 (Disabled)

GUI Tab Advanced tab on page 1006

Location Names

Attribute Location

Description A display of all existing location names defined for the data source you are configuring.

Valid Values *string*

where *string* is the name of a location.

The location names listed in the text box are used for connections according to the order that they are displayed. If you want to change the order or precedence, use the Up and Down buttons.

Default None

GUI Tab General tab on page 1002

Logging

Attribute Logging

Description Creates a log file that logs the SQL execution plan. A value of 0 means no logging is performed.

Valid Values 0 | 1

If set to 1 (Enabled), a log file is created in the current directory. The default log file name is \Integrator.txt.

If set to 0 (Disabled), no logging is performed.

Default 0 (Disabled)

GUI Tab Advanced tab on page 1006

Read Only

Attribute ReadOnly

Description Controls whether the driver opens files for Read-Write access or Read-Only access

Valid Values 0 | 1

If set to 1 (Enabled), the driver opens XML files for Read-Only access. In this case, the XML driver opens XML files with a Shared

Read lock. This allows other connections and applications to read the same XML file that the XML driver has open; however, they cannot write to the XML file.

If set to 0 (Disabled), the XML driver opens XML files for Read-Write access. Opening an XML file for Read-Write access places an exclusive lock on the file. No other connections or applications can open the XML file while the driver has the file open.

Default 0 (Disabled)

GUI Tab Advanced tab on page 1006

Report Codepage Conversion Errors

Attribute ReportCodepageConversionErrors

Description 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

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 on page 1006

Schema Mode

Attribute n/a

Description Specifies whether to show virtual schemas, manufactured schemas, or both.

Valid Values Choose one of the following options:

- Show Virtual Schemas Only. This option returns "XML" in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. See Show Virtual Schemas.

- Show Manufactured Schemas Only. This option returns the manufactured schema names in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. See Show Manufactured Schemas.
- Show Both Virtual and Manufactured Schemas. This option returns both virtual and manufactured schema names when a SQLTables or SQLColumns operation is performed when connected to a data source.

Default Show Virtual Schemas Only

GUI Tab Advanced tab on page 1006

Show Manufactured Schemas

Attribute Show Manufactured Schemas

Description Returns the manufactured schema names in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. The Location names you define for a data source are manufactured into a schema name by adding a # symbol after the Location names. For example:

Schema Name	Table Name
LOC1#	TAB1A
LOC1#	TAB1B
LOC2#	TAB2A
LOC2#	TAB2B

Valid Values 0 | 1

If set to 1 (Enabled), manufactured schema names are returned.

If set to 0 (Disabled), manufactured schema names are not returned.

To return both manufactured and virtual schema names, set this option to 1 (Enabled) and the Show Virtual Schemas option to 1 (Enabled).

Default 0 (Disabled)

GUI Tab Advanced tab on page 1006

Show Virtual Schemas

Attribute Show Virtual Schemas

Description Returns "XML" in the Schema Name column when a SQLTables or SQLColumns operation is performed when connected to a data source. For example:

Schema Name	Table Name
XML	TAB1A
XML	TAB1B
XML	TAB2A
XML	TAB2B

Valid Values 0 | 1

If set to 1 (Enabled), virtual schema names are returned.

If set to 0 (Disabled), virtual schema names are not returned.

To return both virtual and manufactured schema names, set this option to 1 (Enabled) and the Show Manufactured Schemas option to 1 (Enabled).

Default 1 (Enabled)

GUI Tab Advanced tab on page 1006

Configure Location Dialog Box Descriptions

Table 21-3 lists the connection string attributes associated with General and Advanced tabs of the XML driver Configure Location dialog box. The descriptions themselves are listed below the table. See "Defining Locations" on page 996 for an explanation of locations.

NOTE: XML driver connection string attributes do not use short name equivalents.

The names of all connection options in this section are preceded by *location_name*, where *location_name* represents the name of a specific location that you have defined, for example, LOC1. See the description of the Location Name option for details.

Table 21-3. XML Configure Location Attribute Names

Attribute	Default
location_name	None
location_name.Catalog Type Hint	Folder
location_name.Create Type	DataDirect
location_name.Delete Schema	0 (Disabled)
location_name.Flush Every Change	1 (Enabled)
location_name.Initial Catalog	None
location_name.Password	None
location_name.Require Passwd	0 (Disabled)
location_name.Resolve External	0 (Disabled)
location_name.Row Hint	None
location_name.Scan Rows	0
location_name.Table Hint	None
location_name.User ID	None
location_name.Validate Schema	0 (Disabled)

Delete Linked Schema

Attribute *location_name*.Delete Schema

Description Specifies whether an externally-linked schema file is deleted when a table is deleted. This option is valid only for Folder location types. The XML document for the table contains a link to this external schema file. By default, this check box is not selected.

Valid Values 0 | 1

If set to 1 (Enabled), the externally-linked schema file is deleted when the table is deleted. If multiple XML documents are linked to the same schema file, the schema file is not deleted when a table is deleted.

If set to 0 (Disabled), the externally-linked schema file is not deleted when the table is deleted.

Default 0 (Disabled)

GUI Tab Configure Location Advanced tab on page 1004

Flush Every Change

Attribute *location_name*.Flush Every Change

Description Writes the data document to disk after every insert, update, or delete operation. This option is valid only for Folder location types.

Valid Values 0 | 1

If set to 1 (Enabled), the driver writes the data document to disk after every change.

If set to 0 (Disabled), the driver does not write the data document to disk after every change. Disabling this option can improve performance.

Default 1 (Enabled)

GUI Tab Configure Location Advanced tab on page 1004

Location

Attribute *location_name*.Initial Catalog

Description The full path name to the location you are defining.

Valid Values *location_directory*

where *location_directory* is the full path name of the directory in which the data files are stored. For example:

```
LOC1.Initial Catalog=C:\Documents\filesml
```

Default None

GUI Tab Configure Location General tab on page 1003

Location Name

Attribute *location_name*

Description A unique name for the location you are defining, for example, LOC1.

Valid Values *location_name*={DataDirect Closed XML ADO Provider}

where *location_name* is the unique name of the location you are defining. For example, if you choose the location name LOC1, then:

```
LOC1={DataDirect Closed XML ADO Provider}
```

Default None

GUI Tab Configure Location General tab on page 1003

Location Type

Attribute *location_name*.Catalog Type Hint

Description Specifies the type of location you are defining for the connection.

Valid Values Folder | XML Document | HTML Document

For example:

```
LOC1.Catalog Type Hint=XML Document
```

See "Defining Locations" on page 996 for the definition of each type.

Default Folder

GUI Tab Configure Location General tab on page 1003

Max Rows to Scan

Attribute *location_name*.Scan Rows

Description An integer that represents the maximum number of rows to scan when the XML driver is determining the data type of each column. This option is valid only for XML Document location types.

Valid Values 0 | *x*

where *x* is the number of rows to scan.

If set to *x*, the driver scans a maximum of *x* rows in the table. During the scan, the driver inspects each column value in the row of a table and adjusts the data type determination for each column based on the corresponding value. The more sample column values it encounters, the more accurate the determination.

If set to 0, the driver scans all rows in the table. Disabling this option can improve performance because limiting the number of rows can reduce the amount of time it takes to determine the column information on very large documents. Because less information is available, however, the determination of the data types can be incorrect.

Default 0

GUI Tab Configure Location Advanced tab on page 1004

Password

Attribute *location_name*.Password

Description The password used to establish a connection to the location specified by *location_name*. A password is required only if the location to which you are connecting is password-protected.

This option is not available unless the Require User ID/Password option is enabled.

Valid Values *pwd*

where *pwd* is a valid password.

WARNING: The encrypted password is stored in the Windows Registry.

Default None

GUI Tab Configure Location Advanced tab on page 1004

Require User ID/Password

Attribute *location_name*.Require Passwd

Description Specifies whether a User ID and password are required to establish a connection to the location you are defining.

Valid Values 0 | 1

If set to 1 (Enabled), a User ID and password are required to establish a connection to the location. You must enable this option if the location you are defining is password-protected; otherwise, the connection will fail. Enabling this option causes a Logon dialog box to appear when connecting with the driver.

If set to 0 (Disabled), no user ID and password are required to establish a connection to the location.

Default 0 (Disabled)

GUI Tab Configure Location Advanced tab on page 1004

Resolve External References

Attribute *location_name*.Resolve External

Description Determines whether external references such as DTDs, Schemas, Entities, and Notations are resolved for the XML documents contained within the location specified by *location_name*.

Valid Values 0 | 1

If set to 1 (Enabled), the documents are not processed if the XML parser cannot locate the external references.

If set to 0 (Disabled), the document is processed, even if the XML parser cannot locate the external references.

Default 0 (Disabled)

GUI Tab Configure Location Advanced tab on page 1004

Row Hint

Attribute *location_name*.Row Hint

Description A string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the nodes that make up the rows in the rowset of a tabular-formatted XML document contained within the location specified by *location_name*. See "Using Hints for Tabular-Formatted XML Documents" on page 1028 for details.

This option is valid only for Folder and HTML Document location types.

Valid Values *row_hint*

where *row_hint* is an XSL pattern.

Default None

GUI Tab Configure Location Advanced tab on page 1004

Table Creation

Attribute *location_name*.Create Type

Description Determines the style of XML that is generated when a new table is created. This option is valid only for Folder location types.

Valid Values IE5DataIsland | ADO25 | DataDirect

- Data Island Format (IE5DataIsland): New tables are created with the Internet Explorer 5 Data Island XML style.
- ADO Format (ADO25): New tables are created with the ADO 2.5 XML style.
- DataDirect Format (DataDirect): New tables are created with the DataDirect format. This format conforms to the W3C recommendation for XML schema, working draft April 07, 2000.

See "Common Tabular Formats for XML Documents" on page 992 for a description of each of these formats.

Default DataDirect

GUI Tab Configure Location Advanced tab on page 1004

Table Hint

Attribute *location_name*.Table Hint

Description A string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the table or rowset nodes in a tabular-formatted XML document contained within the location specified by *location_name*. See "Using Hints for Tabular-Formatted XML Documents" on page 1028 for details.

This option is valid only for Folder and HTML Document location types.

Valid Values `table_hint`

where `table_hint` is an XSL pattern.

Default None

GUI Tab Configure Location Advanced tab on page 1004

User ID

Attribute *location_name*.User ID

Description The User ID (user name) used to establish a connection to the location specified by *location_name*. A password is required only if the location to which you are connecting is password-protected.

This option is not available unless the Require User ID/Password option is enabled.

Valid Values `userid`

where `userid` is a valid user name.

Default None

GUI Tab Configure Location Advanced tab on page 1004

Validate Schema

Attribute *location_name*.Validate Schema

Description Determines whether the XML documents contained within the location specified by *location_name* are validated against their schema.

Valid Values 0 | 1

If set to 1 (Enabled), the XML documents are validated against their schema. This allows a well-formed XML document to be processed, even if the document is not valid.

If set to 0 (Disabled), the XML documents are not validated against their schema.

Default 0 (Disabled)

GUI Tab Configure Location Advanced tab on page 1004

Using Hints for Tabular-Formatted XML Documents

The XML driver supports table and row hints. You can specify a table hint, a row hint, or both, when configuring an XML data source or using a connection string.

Table hints should be specified so that they resolve to a single node. If a table hint resolves to a set of nodes, the first node in the set is used as the table node. The context of the table hint is always the root node of the XML document.

Row hints define the "row" element and specify whether the rowset is element-based or attribute-based. If a table hint is supplied, the context of the row node is the node to which the table hint resolves; otherwise, the context is the root node of the XML document. The column mode identifier specifies whether the columns of a row are child nodes or attributes of the row node.

When working with hints, keep in mind that the XML driver assumes that the row nodes are the immediate children of the table node.

- If only a table hint is specified, the row nodes are the children of the node to which the hint resolves. It is assumed that all of the child nodes have the same name.
- If only a row hint is specified, the table node is the parent of the node to which the hint resolves. If the row hint resolves to a set of nodes, the nodes in that set must all have the same parent.
- If both a table hint and a row hint are specified, the row hint is taken to be relative to the node to which the table hint resolves.

The column mode identifier has the format:

`\column` *`mode`*

where *`mode`* can be one of the following options:

- child: The columns are child nodes of the row node.
- attr: The columns are attributes of the row node.

In the following examples, the columns are the children of the row nodes.

Example 1

Table Hint:

Row Hint: //Item

The row nodes are the nodes named Item. The table node is the parent of the row nodes. Use this form only when all of the Item nodes reside under one parent.

If some Item nodes have different parents, use a table hint or a more specific row hint to select the set of Item nodes.

Example 2

Table Hint:

Row Hint: /Bookstore/Books/Item

The row nodes are the nodes named Item. The table node is Books, which is a child of the Bookstore node.

Example 3

Table Hint: /Bookstore/Books

Row Hint:

The table node is Books, which is a child of the Bookstore node. The row nodes are the children of the Books node. It is assumed that all of the child nodes under the Books nodes have the same name. If the child nodes do not all have the same name, the name of the first child node encountered is used as the row node name. In that case, it would be better to specify both a table and row hint.

Example 4

Table Hint: /Bookstore [@location = "Raleigh"]/Books

Row Hint: ./Item

The table node is Books, which is a child of the Bookstore node. Bookstore has a "location" attribute with the value Raleigh. The row nodes are the Item nodes that are children of the Books node.

Column Mode Identifier

The following examples illustrate the use of the optional column mode identifier.

Example 5

Table Hint:

Row Hint: //Item \column attr

The row nodes are named Item. The table node is the parent of the row nodes. The columns are attributes of the row node.

Example 6

Table Hint:

Row Hint: //Item \column child

The row nodes are the nodes named Item. The table node is the parent of the row nodes. The columns are attributes of the row node.

Data Types

This section provides three tables that show how the data types for each supported tabular-formatted XML document map to the standard ODBC data types, as follows:

- Table 21-4, “Data Islands Data Types,” on page 1032
- Table 21-5, “ADO 2.5 Persisted Files Data Types,” on page 1033
- Table 21-6, “DataDirect Format Data Types,” on page 1034

Table 21-4. Data Islands Data Types

Data Islands	Internal XML Name	ODBC
binhex	bin.hex	SQL_LONGVARBINARY
boolean	boolean	SQL_BIT
currency	fixed.14.4	SQL_DECIMAL
date	date	SQL_TYPE_DATE
dateTime	dateTime	SQL_TYPE_TIMESTAMP
float	float	SQL_DOUBLE
i1	i1	SQL_TINYINT SIGNED
i2	i2	SQL_SMALLINT SIGNED
i4	i4	SQL_INTEGER SIGNED
int	int	SQL_INTEGER SIGNED
number	number	SQL_DOUBLE
r4	r4	SQL_REAL
r8	r8	SQL_DOUBLE
singleChar	singleChar	SQL_SMALLINT
string	string	SQL_WLONGVARCHAR
time	time	SQL_TYPE_TIME
ui1	ui1	SQL_TINYINT UNSIGNED
ui2	ui2	SQL_SMALLINT UNSIGNED
ui4	ui4	SQL_INTEGER UNSIGNED

Table 21-5. ADO 2.5 Persisted Files Data Types

ADO 2.5 Persisted Files	Internal XML Name	ODBC
binhex	bin.hex	SQL_LONGVARBINARY
boolean	boolean	SQL_BIT
currency	fixed.14.4	SQL_DECIMAL
date	date	SQL_TYPE_DATE
dateTime	dateTime	SQL_TYPE_TIMESTAMP
float	float	SQL_DOUBLE
i1	i1	SQL_TINYINT SIGNED
i2	i2	SQL_SMALLINT SIGNED
i4	i4	SQL_INTEGER SIGNED
i8	i8	SQL_BIGINT SIGNED
int	int	SQL_INTEGER UNSIGNED
number	number	SQL_DOUBLE
r4	r4	SQL_REAL
r8	r8	SQL_DOUBLE
singleChar	singleChar	SQL_SMALLINT SIGNED
time	time	SQL_TYPE_TIME
ui1	ui1	SQL_TINYINT UNSIGNED
ui2	ui2	SQL_SMALLINT UNSIGNED
ui4	ui4	SQL_INTEGER UNSIGNED
ui8	ui8	SQL_BIGINT UNSIGNED
wchar	string	SQL_CHAR
wchar	string	SQL_WCHAR
wlvarchar	string	SQL_WLONGVARBINARY
wvarchar	string	SQL_WVARCHAR

Table 21-6. DataDirect Format Data Types

DataDirect	Internal XML Name	ODBC
binary	binary	SQL_BINARY
boolean	boolean	SQL_BIT
byte	byte	SQL_TINYINT SIGNED
date	date	SQL_TYPE_DATE
decimal	decimal	SQL_NUMERIC
double	double	SQL_DOUBLE
float	float	SQL_REAL
int	int	SQL_INTEGER UNSIGNED
long	long	SQL_BIGINT SIGNED
lvarbinary	binary	SQL_LONGVARBINARY
short	short	SQL_SMALLINT SIGNED
time	time	SQL_TYPE_TIME
timeInstant	timeInstant	SQL_TYPE_TIMESTAMP
unsignedByte	unsignedByte	SQL_TINYINT UNSIGNED
unsignedInt	unsignedInt	SQL_INTEGER UNSIGNED
unsignedLong	unsignedLong	SQL_BIGINT UNSIGNED
unsignedShort	unsignedShort	SQL_SMALLINT UNSIGNED
varbinary	binary	SQL_VARBINARY
wchar	string	SQL_CHAR
wchar	string	SQL_WCHAR
wlvarchar	string	SQL_WLONGVARBINARY
wvarchar	string	SQL_WVARCHAR

See "Retrieving Data Type Information" on page 77 for information about retrieving data types.

Unicode Support

The driver supports the Unicode ODBC W (Wide) function calls, such as SQLConnectW. This allows the Driver Manager to transmit these calls directly to the driver. Otherwise, the Driver Manager would incur the additional overhead of converting the W calls to ANSI function calls, and vice versa.

See "UTF-16 Applications on UNIX and Linux" on page 168 for related details. Also, Refer to Chapter 4 "Internationalization, Localization, and Unicode" in the *DataDirect Connect Series for ODBC Reference* for a more detailed explanation of Unicode.

Persisting a Result Set as an XML Data File

The driver allows you to persist a result as an XML data file with embedded schema. See "Persisting a Result Set as an XML Data File" on page 79 for details about implementation.

ODBC Conformance Level

The XML driver supports SQLSetPos.

Refer to Chapter 2 "ODBC API and Scalar Functions" in the *DataDirect Connect Series for ODBC Reference* for a list of the API functions supported by the XML driver.

Number of Connections and Statements Supported

There is no limit to the number of connections and statements supported.

SQL Supported

This section provides information about the SQL statements that the XML driver processes, and about SQL standards and conventions that the driver supports:

- "SQL Statements" on page 1036
- "Extensions to SQL Standards" on page 1037
- "Grammar Token Definitions" on page 1037

SQL Statements

The SQL Engine included with the XML driver supports the following SQL statements:

- Select
- Create and Drop Table
- Insert
- Update
- Delete

NOTE: See the table at the beginning of this chapter for the SQL statements that the XML driver supports for the different types of supported file formats.

Extensions to SQL Standards

The XML driver uses SQL grammar that is compliant with entry level ANSI SQL-92. Table 21-7 summarizes significant extensions to the grammar.

Table 21-7. SQL Extensions

Entry Level ANSI SQL-92 Extension	Relevant Standard or Convention
Aliasing table references	Intermediate level ANSI SQL-92
ANSI date, time, and timestamp literals	Intermediate level ANSI SQL-92
Dynamic parameter specification	Full level ANSI SQL-92
GUID literals	COM
Hex string literals	Full level ANSI SQL-92
Left Outer Joins	Intermediate level ANSI SQL-92
ODBC escape support	ODBC 3.0
Scalar functions	ODBC 3.0

Grammar Token Definitions

The tokens used in the XML driver SQL grammar are defined in the following sections:

- "Regular Identifiers" on page 1038
- "Delimited Identifiers" on page 1038
- "Integer Numbers" on page 1038
- "Real Numbers" on page 1039
- "Character String Literals" on page 1039
- "GUID Literals" on page 1039
- "Hex Literals" on page 1040
- "Time and Date Literals" on page 1040

- "SQL Operators and Symbols" on page 1041
- "Keywords for the XML Driver" on page 1041
- "SQL Comments" on page 1046

Regular Identifiers

A regular identifier must begin with a letter and may not exceed 128 characters. In addition, all ASCII characters are converted to uppercase.

The following are examples of regular identifiers:

- FOO
- COLUMN_NAME
- SCHEMA#NAME
- Col3 (legal, but converted to COL3)

Delimited Identifiers

Delimited identifiers may not exceed 128 characters. A double quotation character can be embedded within the string by specifying two consecutive double quotation mark characters. A delimited identifier can span multiple lines. The body of a delimited identifier can contain any character except the newline character.

The following examples show delimited identifiers:

- "$ % ^ ($"
- "This is a delimited variable name"

Integer Numbers

Examples of integer numbers are:

- 5
- 1004

Real Numbers

Examples of real numbers are:

- .10
- 12.01
- 10.
- .01e-10
- 12E+10
- 12.01e2
- 12.01e-10
- 12.e-10

Character String Literals

Character string literals are delimited with single quotation mark characters. A single quotation mark character can be embedded within the string by specifying two consecutive single quotation mark characters. A character string literal can span multiple lines.

Examples are:

- '$%^($'
- 'This is a character string literal’

GUID Literals

A GUID uses the following format, where *x* is a hexadecimal digit:

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Hex Literals

Hex literal values are introduced with an uppercase `X` followed by a single quoted string of hexadecimal characters.

Examples are:

- `X'39FA'`
- `X'B0F00D'`

Time and Date Literals

Date, time, and timestamp literals are date, time, and timestamp values surrounded by a standard prefix and suffix. Date literals are specified in a *YYYY-MM-DD* format. Time literals are specified in an *HH:MM:SS* format with an optional fraction component. Timestamp literals are a concatenation of date and time values.

Examples for ODBC and SQL syntax are shown in the following table.

Literal Type	ODBC Syntax	ANSI SQL-92 Syntax
Date Literal	`{d '1999-09-19'}`	`date '1999-09-19'`
Time Literal	`{t '11:11:11.225'}`	`time '11:11:11.225'`
Timestamp Literal	`{ts '1999-09-19 11:11:11.225'}`	`timestamp '1999-09-19 11:11:11.225'`
Timestamp Literal	`{ts '1999-09-19'}`	`timestamp '1999-09-19'`

NOTE: ODBC 1.*x* style ODBC escape sequences such as the following are not supported:

```
--(*VENDOR(Microsoft), PRODUCT(ODBC) ...*)--
```

SQL Operators and Symbols

Symbol	Description	Symbol	Description
':'	Colon	'<'	Less than operator
';'	Semicolon	‘)’	Right parenthesis
'.'	Period	‘=’	Equal operator
','	Comma	‘+’	Plus operator
'<>'	Not equal operator	‘-’	Minus operator
'<='	Less than or equal operator	‘*’	Multiply operator
'>='	Greater than or equal operator	‘/’	Divide operator
‘>’	Greater than operator	‘?’	Dynamic parameter
‘(‘	Left parenthesis		

Keywords for the XML Driver

A keyword may not be used as a regular identifier. For example, the following statement would generate a syntax error because INDICATOR is a keyword:

```
SELECT INDICATOR FROM T1
```

You can, however, enclose a keyword in double quotation marks to form a delimited identifier. For example, the following statement is valid:

```
SELECT "INDICATOR" FROM T1
```

Table 21-8 lists all of the keywords that are reserved for use in SQL statements or designated as potential future reserved words.

Table 21-8. Reserved Keywords

ABSOLUTE	ACTION	ADD
AFTER	ALIAS	ALL
ALLOCATE	ALTER	AND
ANY	ARE	AS
ASC	ASSERTION	ASYNC
AT	AUTHORIZATION	AVG
BEFORE	BEGIN	BETWEEN
BIT	BIT_LENGTH	BOOLEAN
BOTH	BREADTH	BY
CALL	CASCADE	CASCADED
CASE	CAST	CATALOG
CHAR	CHAR_LENGTH	CHARACTER
CHARACTER_ LENGTH	CHECK	CLOSE
COALESCE	COLLATE	COLLATION
COLUMN	COLUMNS	COMMIT
COMPLETION	CONCAT	CONNECT
CONNECTION	CONSTRAINT	CONSTRAINTS
CONTINUE	CONVERT	CORRESPONDING
COUNT	CREATE	CROSS
CURDATE	CURRENT	CURRENT_DATE
CURRENT_TIME	CURRENT_ TIMESTAMP	CURRENT_USER
CURSOR	CURTIME	CYCLE
DATA	DATE	DAY
DAYOFMONTH	DAYOFWEEK	DEALLOCATE
DEC	DECIMAL	DECLARE
DEFAULT	DEFERRABLE	DEFERRED
DELETE	DEPTH	DESC
DESCRIBE	DESCRIPTOR	DIAGNOSTICS

Table 21-8. Reserved Keywords *(cont.)*

DICTIONARY	DISCONNECT	DISTINCT
DOMAIN	DOUBLE	DROP
EACH	ELSE	ELSEIF
END	END_EXEC	EQUALS
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FLOOR	FOR	FOREIGN
FOUND	FROM	FULL
GENERAL	GET	GLOBAL
GO	GOTO	GRANT
GROUP	HAVING	HOUR
IDENTIFY	IF	IFNULL
IGNORE	IMMEDIATE	IN
INDEX	INFO	INDICATOR
INITIALLY	INNER	INPUT
INSENSITIVE	INSERT	INT
INTEGER	INTERSECT	INTERVAL
INTO	IS	ISOLATION
JOIN	KEY	LANGUAGE
LAST	LCASE	LEADING
LEAVE	LEFT	LENGTH
LESS	LEVEL	LIKE
LIMIT	LOCAL	LOOP
LOWER	LTRIM	MATCH
MAX	MIN	MINUTE
MOD	MODIFY	MODULE
MONTH	NAMES	NATIONAL
NATURAL	NCHAR	NEW

Table 21-8. Reserved Keywords *(cont.)*

NEXT	NO	NONE
NOT	NOW	NULL
NULLIF	NUMERIC	OBJECT
OCTET_LENGTH	OF	OFF
OID	OLD	ON
ONLY	OPEN	OPERATION
OPERATORS	OPTION	OR
ORDER	OTHERS	OUTER
OUTPUT	OVERLAPS	PAD
PARAMETERS	PARTIAL	PENDANT
POSITION	POWER	PRECISION
PREORDER	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVATE
PRIVILEGES	PROCEDURE	PROTECTED
PUBLIC	RCASE	READ
REAL	RECURSIVE	REF
REFERENCES	REFERENCING	RELATIVE
REMOVE	REPLACE	RESIGNAL
RESTRICT	RETURN	RETURNS
REVOKE	RIGHT	ROLE
ROLLBACK	ROUND	ROUTINE
ROW	ROWS	RTRIM
SAVEPOINT	SCHEMA	SCROLL
SEARCH	SECOND	SECTION
SELECT	SENSITIVE	SEQUENCE
SESSION	SESSION_USER	SET
SIGNAL	SIMILAR	SIZE
SMALLINT	SOME	SPACE
SQL	SQLCODE	SQLERROR
SQLEXCEPTION	SQLSTATE	SQLWARNING

Table 21-8. Reserved Keywords *(cont.)*

STRUCTURE	SUBSTRING	SUM
SYSTEM_USER	TABLE	TEMPORARY
TEST	THEN	THERE
TIME	TIMESTAMP	TIMEZONE_ HOUR
TIMEZONE_ MINUTE	TO	TRAILING
TRANSACTION	TRANSLATE	TRANSLATION
TRIGGER	TRIM	TRUE
TYPE	UCASE	UNDER
UNION	UNIQUE	UNKNOWN
UPDATE	UPPER	USAGE
USER	USING	VALUE
VALUES	VARCHAR	VARIABLE
VARYING	VIEW	VIRTUAL
VISIBLE	WAIT	WHEN
WHENEVER	WHERE	WHILE
WITH	WITHOUT	WORK
WRITE	YEAR	ZONE

SQL Comments

ANSI SQL-92 standard comments (--) and C++ standard comments (/*...*/, //) are supported. Comments can be nested.

For example, in the following query columns col2, col3, and col4 are ignored:

```
SELECT col1  /* col1 comment */
/*
    col2,    -- col2 comment
    col3,    // col3 comment
    col4,    /* col4 comment */
*/
FROM t1
```

Index

Symbols

symbol in regular identifiers 1038

A

about
  connection failover 87
  connection pooling 109
  DataDirect Bulk Load 115
  extended connection failover 89
  select failover 92
Account String, Teradata 807
Accounting Info, DB2 Wire Protocol 211
Action for Undefined Tables
  Btrieve 843
  Text 968
Add to Create Table, DB2 Wire Protocol 212
adding connections to a connection pool 110
address, IP 70
Administrator, Data Source
  UNIX and Linux 46, 138
  Windows 42
ADO 2.5 persisted file format for XML
  driver 992
AIX
  See UNIX and Linux
aliasing table references 1037
Allow Update and Delete, Text driver 969
Alter Table statement
  Btrieve 856
  dBASE 890
  Paradox 945
  Text 989
alternate database servers, about 87
Alternate ID, DB2 Wire Protocol 212
alternate server, specifying
  guidelines for 94
Alternate Servers
  DB2 Wire Protocol 213
  Greenplum Wire Protocol 278
  Informix Wire Protocol 321
  MySQL Wire Protocol 354
  Oracle 714
  Oracle Wire Protocol 418
  PostgreSQL Wire Protocol 498
  SQL Server Classic Wire Protocol 766
  SQL Server Native Wire Protocol 564
  Sybase Wire Protocol 641
AnsiNPW
  SQL Server Classic Wire Protocol 766
  SQL Server Native Wire Protocol 566
ansinull option, Sybase Wire Protocol 690
Application Name
  DB2 Wire Protocol 214
  SQL Server Classic Wire Protocol 767
  SQL Server Native Wire Protocol 565, 566
  Sybase Wire Protocol 642
Application Using Threads
  Btrieve 844
  DB2 Wire Protocol 214
  dBASE 874
  Greenplum Wire Protocol 279
  Informix 913
  Informix Wire Protocol 322
  MySQL Wire Protocol 355
  Oracle 715
  Oracle Wire Protocol 419
  Paradox 938
  PostgreSQL Wire Protocol 499
  SQL Server Native Wire Protocol 567
  Sybase Wire Protocol 642
  Text 969

array binding
  Oracle Wire Protocol 477
  use in bulk load operations 128
Array Size
  Btrieve 844
  Oracle 715
  Oracle Wire Protocol 420
  Sybase Wire Protocol 658
arrays of parameters
  DB2 Wire Protocol 260
  Greenplum Wire Protocol 307
  Oracle 750
  Oracle Wire Protocol 479
  PostgreSQL Wire Protocol 533
  SQL Server Classic Wire Protocol 789
  SQL Server Native Wire Protocol 606
  Sybase Wire Protocol 694
AttachDBFileName, SQL Server Classic Wire Protocol 767
authentication
  SSL client 107
  SSL server 106
  user 102
Authentication Method
  DB2 Wire Protocol 215
  Oracle Wire Protocol 420
  SQL Server Native Wire Protocol 567
  Sybase Wire Protocol 643
AuthenticationDomain, Teradata 819
AuthenticationPassword, Teradata 807
AuthenticationUserID, Teradata 808
AutoTranslate
  SQL Server Classic Wire Protocol 768

B

Batch Size
  DB2 Wire Protocol 216
  Oracle Wire Protocol 421
  SQL Server Native Wire Protocol 568
  Sybase Wire Protocol 644
bind packages, DB2 Wire Protocol 174
binding dynamic parameters 72
binding SQL statements 72
Btrieve driver
  Alter Table statement 856
  column names 854
  connection string attributes 843
  connections supported 859
  Create Index statement 857
  data dictionary file 835
  data source
    configuring 836
    connecting via connection string 841
  data types 852
  driver requirements 834
  Drop Index statement 857
  indexes 854
  isolation levels 858
  locking levels 858
  managing databases 835
  ODBC conformance 858
  Rowid pseudo-column 855
  Scalable SQL 835
  Select statement 855
  statements supported 859
  table structure 841
  transactions 835
Bulk Binary Threshold
  DB2 Wire Protocol 216
  Oracle Wire Protocol 422
  SQL Server Native Wire Protocol 569
  Sybase Wire Protocol 644
Bulk CharacterThreshold
  DB2 Wire Protocol 217
  Oracle Wire Protocol 422
  SQL Server Native Wire Protocol 569
  Sybase Wire Protocol 645
bulk load
  bulk data configuration file 124
  configuring on Sybase 691
  determining the protocol 128
  external overflow file 129
  overview 115
  sample application 127

transactions on Sybase 691
validating files 123
bulk load data file
setting the maximum size of binary data exported
DB2 Wire Protocol 216
Oracle Wire Protocol 422
SQL Server Native Wire Protocol 569
Sybase Wire Protocol 644
setting the maximum size of character data exported
DB2 Wire Protocol 217
Oracle Wire Protocol 422
SQL Server Native Wire Protocol 569
Sybase Wire Protocol 645
Bulk Options, SQL Server Native Wire Protocol 570
BulkBinaryThreshold
DB2 Wire Protocol 216
Oracle Wire Protocol 422
SQL Server Native Wire Protocol 569
Sybase Wire Protocol 644
BulkLoadBatchSize
DB2 Wire Protocol 216
Oracle Wire Protocol 421
SQL Server Native Wire Protocol 568
Sybase Wire Protocol 644

C

Cache Size
dBASE 875
Text 970
Cached Cursor Limit, Oracle Wire Protocol 423
Cached Description Limit, Oracle Wire Protocol 424
Cancel Detect Interval
Informix 914
Informix Wire Protocol 323
Catalog Functions Include Synonyms
Oracle 716
Oracle Wire Protocol 424
Catalog Options
Oracle 716
Oracle Wire Protocol 425
Catalog Schema, DB2 Wire Protocol 218
Century Boundary, Text 970
character set conversions 128
Character Set for CCSID 65535, DB2 Wire Protocol 218
character string literals 1039
characters, unexpected
DB2 Wire Protocol 255
Oracle 742
Oracle Wire Protocol 470
Sybase Wire Protocol 686
Charset, Sybase Wire Protocol 646
Client Host Name, DB2 Wire Protocol 219
client load balancing, about 95
Client User, DB2 Wire Protocol 219
Client Version, Oracle 716, 717
code pages, IBM DB2 253
Collection, DB2 Wire Protocol 220
Column Names in First Line, Text 971
comma-separated value (CSV) format file
character set conversions 128
use in bulk load 124
comments, SQL 1046
configuration file, bulk data 124
Connection Cache Size, Sybase Wire Protocol 647
connection failover
about 85, 87
connection retry and 88
DB2 Wire Protocol 259
Greenplum Wire Protocol 305
Informix Wire Protocol 334
load balancing and 95
MySQL Wire Protocol 382
Oracle 748
Oracle Wire Protocol 477
PostgreSQL Wire Protocol 531
SQL Server Classic Wire Protocol 787

SQL Server Native Wire Protocol 603
Sybase Wire Protocol 692
connection pool
adding connections 110
creating 110
dead connections 112
removing connections 111
when a physical connection to a server is lost 112
Connection Pooling
DB2 Wire Protocol 221
Greenplum Wire Protocol 279
MySQL Wire Protocol 355
Oracle Wire Protocol 425
PostgreSQL Wire Protocol 500
SQL Server Native Wire Protocol 571
Sybase Wire Protocol 647
Connection Reset
DB2 Wire Protocol 221
Greenplum Wire Protocol 280
MySQL Wire Protocol 355, 356
Oracle Wire Protocol 425, 426
PostgreSQL Wire Protocol 500
SQL Server Native Wire Protocol 572
Sybase Wire Protocol 648
Connection Retry Count
DB2 Wire Protocol 222
Greenplum Wire Protocol 281
Informix Wire Protocol 323
MySQL Wire Protocol 357
Oracle 718
Oracle Wire Protocol 427
PostgreSQL Wire Protocol 501
SQL Server Classic Wire Protocol 769
SQL Server Native Wire Protocol 573
Sybase Wire Protocol 649
Connection Retry Delay
DB2 Wire Protocol 222
Greenplum Wire Protocol 281
Informix Wire Protocol 324
MySQL Wire Protocol 357
Oracle 718
Oracle Wire Protocol 427
PostgreSQL Wire Protocol 502
SQL Server Classic Wire Protocol 770
SQL Server Native Wire Protocol 573
Sybase Wire Protocol 649
connection retry, about 88, 96
connection string attributes
Btrieve 843
DB2 Wire Protocol 209
dBASE 874
Greenplum Wire Protocol 277
Informix 912
Informix Wire Protocol 320
MySQL Wire Protocol 353
Oracle 713
Oracle Wire Protocol 416
Paradox 937
PostgreSQL Wire Protocol 497
SQL Server Classic Wire Protocol
UNIX and Linux 765
Windows 763
SQL Server Native Wire Protocol 562
Sybase Wire Protocol 639
Teradata 806
Text 967
XML 1011, 1019
connections supported
Btrieve 859
DB2 Wire Protocol 260
dBASE 898
Greenplum Wire Protocol 306
Informix 928
Informix Wire Protocol 335
MySQL Wire Protocol 383
Oracle 749
Oracle Wire Protocol 479
Paradox 955
PostgreSQL Wire Protocol 532
SQL Server Classic Wire Protocol 789
SQL Server Native Wire Protocol 605
Sybase Wire Protocol 694
Teradata 829
Text 990
XML 1036
contacting Technical Support 35
conventions, typographical 30

Create Index statement
Btrieve 857
dBASE 891
Paradox 952
Create Table statement, Paradox 946
Create Type
dBASE 876
Paradox 938
CSV
See comma-separated value
Current Function Path, DB2 Wire Protocol 223
CursorBehavior, Informix 915
CursorCacheSize, Sybase Wire Protocol 647

D

data encryption 103
Data File Extension
Btrieve 845
dBASE 876
Text 972
Data Island format for XML driver 993
data source
configuring 338
Btrieve 836
DB2 Wire Protocol 181
dBASE 862
FoxPro DBC 868
Greenplum Wire Protocol 264
Informix 901
Informix Wire Protocol 310
Oracle Wire Protocol 387, 482, 699
Paradox 931
SQL Server Classic Wire Protocol 752
SQL Server Native Wire Protocol 537
Sybase Wire Protocol 609
Teradata 792
Text 959
UNIX and Linux 44, 137
Windows 42
XML 1000
connecting via connection string
Btrieve 841
DB2 Wire Protocol 205
dBASE 872, 965
Greenplum Wire Protocol 274
Informix 908
Informix Wire Protocol 317
MySql Wire Protocol 350
Oracle 710
Oracle Wire Protocol 412
Paradox 935
PostgreSQL Wire Protocol 494
SQL Server Classic Wire Protocol 759
SQL Server Native Wire Protocol 559
Sybase Wire Protocol 635
Teradata 802
XML 1008
connecting via logon dialog box
DB2 Wire Protocol 206
Greenplum Wire Protocol 275
Informix 910
Informix Wire Protocol 319
MySQL Wire Protocol 351
Oracle 711
Oracle Wire Protocol 413
PostgreSQL Wire Protocol 495
SQL Server Classic Wire Protocol 761
SQL Server Native Wire Protocol 560
Sybase Wire Protocol 637
Teradata 803
XML 1010
Data Source Administrator
UNIX and Linux 46, 138
Windows 42
Data Source Name
Btrieve 845
DB2 Wire Protocol 224
dBASE 877
Greenplum Wire Protocol 282
Informix 916
Informix Wire Protocol 325
MySQL Wire Protocol 358
Oracle 719
Oracle Wire Protocol 428

PostgreSQL Wire Protocol 502
SQL Server Classic Wire Protocol 770
SQL Server Native Wire Protocol 574
Sybase Wire Protocol 650
Teradata 808
Text 972
XML 1012
data types
Btrieve 852
DB2 Wire Protocol 253
dBASE 887
Greenplum Wire Protocol 300
Informix 923
Informix Wire Protocol 332
MySQL Wire Protocol 380
Oracle 737
Oracle Wire Protocol 466
Paradox 943
PostgreSQL Wire Protocol 526
SQL Server Classic Wire Protocol 784
SQL Server Native Wire Protocol 599
Sybase Wire Protocol 684
Teradata 826
Text 988
XML 1031
Database
Paradox 940
Database Directory
Btrieve 846
dBASE 877
Paradox 940
Text 973
Database Directory, Btrieve 846
Database List
Informix 915
Sybase Wire Protocol 651
Database Name
DB2 Wire Protocol 224
dBASE 878
Greenplum Wire Protocol 282
Informix 916
Informix Wire Protocol 325
MySQL Wire Protocol 358
Paradox 939
PostgreSQL Wire Protocol 503
SQL Server Classic Wire Protocol 770
SQL Server Native Wire Protocol 574
Sybase Wire Protocol 651
DataDirect Bulk Load
character set conversions 128
DB2 Wire Protocol 258
external overflow file 129
Oracle Wire Protocol limitations 477
overview 115
Sybase Wire Protocol, configuring 691
DataDirect Connect for ODBC drivers
Btrieve 833
DB2 Wire Protocol 173
dBASE 861
Greenplum Wire Protocol 263
Informix 899
Informix Wire Protocol 309
MySQL Wire Protocol 337
Oracle 695
Oracle Wire Protocol 385
Paradox 929
PostgreSQL Wire Protocol 481
SQL Server Classic Wire Protocol 751
SQL Server Native Wire Protocol 535
Sybase Wire Protocol 607
Text 957
XML 991
DataDirect Connect XE and Connect64 XE for ODBC drivers, Teradata 791
DataDirect Connect64 for ODBC drivers
DB2 Wire Protocol 173
Informix Wire Protocol 309
MySQL Wire Protocol 337
Oracle 695
Oracle Wire Protocol 385
PostgreSQL Wire Protocol 481
SQL Server Classic Wire Protocol 751
SQL Server Native Wire Protocol 535
date and time literals 1040
date masks, defining for the Text 986

DB2
  connection properties
    for Bulk Load 130
    for connection failover 97
    for connection pooling 114
    for security 108
  High Availability Disaster Recovery (HADR) 94
  IBM code page values 253
DB2 Wire Protocol driver
  authentication 188
  bind packages 174
  bulk load operations 258
  connection failover 259
  connection pool
    creating 110
    dead connections 112
  connection string attributes 209
  connections supported 260
  data encryption 188
  data source
    configuring 181
    connecting via connection string 205
    connecting via logon dialog box 206
  data types 253
  driver requirements 174
  isolation levels 259
  Kerberos authentication 177
  locking levels 259
  ODBC conformance 260
  persisting result set as XML 259
  statements supported 260
  stored procedure support 258
  unexpected characters 255
  Unicode support 255
  Workload Manager (WLM) service class 178, 252
  workloads
    overview 178
    performance 179
  XML data type 254
  XQuery expressions 258
dBASE driver
  Alter Table statement 890
  column names 889
  connection string attributes 874
  connections supported 898
  Create Index statement 891
  data source
    configuring 862
    connecting via connection string 872
  data types 887
  defining index attributes 884
  defining index attributes under UNIX 886
  driver requirements 861
  Drop Index statement 893
  isolation levels 897
  locking 896
  locking levels 897
  ODBC conformance 898
  Pack statement 894
  Rowid pseudo-column 889
  Select statement 889
  statements supported 898
DBCName List, Teradata 809
DBCName or Alias, Teradata 809
ddtestlib tool 136
dead connections in a connection pool 112
Decimal Symbol, Text 973
dedicated bulk protocol 128
Default Buffer Size for Long/LOB Columns
  MySQL Wire Protocol 359
  Oracle 719
  Oracle Wire Protocol 428
  Sybase Wire Protocol 652
Default Database, Teradata 811
Default Isolation Level, DB2 Wire Protocol 224
Default Logon ID, Btrieve 846
Default Role, Teradata 811
Default Table Type, Text 974
Delete Linked Schema, XML 1020
Delimiter Character, Text 974
demo tool, XML persistence
  DB2 Wire Protocol 259
  Greenplum Wire Protocol 305
  Informix 927
  Informix Wire Protocol 334

MySQL Wire Protocol 382
Oracle 748
Oracle Wire Protocol 477
PostgreSQL Wire Protocol 531
SQL Server Native Wire Protocol 603
Sybase Wire Protocol 693
Teradata 829
XML 1035
demoodbc application 160
Describe at Prepare
Oracle 720
Oracle Wire Protocol 429
Distributed Transaction Coordinator
Informix Wire Protocol 334
Oracle 744
Oracle Wire Protocol 473
Sybase Wire Protocol 689
Distributed Transaction Model, Sybase Wire Protocol 653
distributed transactions
Informix 925
Informix Wire Protocol 334
Oracle 744
Oracle Wire Protocol 473
Sybase Wire Protocol 689
documentation, about 32
Driver Options, XML 1012
driver requirements
Btrieve 834
DB2 Wire Protocol 174
dBASE 861
Greenplum Wire Protocol 263
Informix 900
Informix Wire Protocol 309
MySQL Wire Protocol 338
Oracle 695
Oracle Wire Protocol 385
Paradox 929
PostgreSQL Wire Protocol 481
SQL Server Classic Wire Protocol 751
SQL Server Native Wire Protocol 535
Sybase Wire Protocol 607
Teradata 791
Text 957
XML 992
drivers
about 58
version string information 74
Drop Index statement
Btrieve 857
dBASE 893
Paradox 953
DTC
See distributed transactions
dynamic parameters, binding 72
Dynamic Sections 226

E

Enable Bulk Load
DB2 Wire Protocol 227
Oracle Wire Protocol 430
SQL Server Native Wire Protocol 575
Sybase Wire Protocol 653
Enable Data Encryption, Teradata 812
Enable Describe Parameter 654
Enable Extended Statement Information, Teradata 812
Enable Insert Cursors, Informix 917
Enable LOBs, Teradata 813
Enable N-CHAR Support
Oracle 721
Oracle Wire Protocol 430
Enable Quoted Identifiers
SQL Server Classic Wire Protocol 772
SQL Server Native Wire Protocol 575
Sybase Wire Protocol 654
Enable Reconnect, Teradata 814
Enable Scrollable Cursors
Oracle 722
Oracle Wire Protocol 431
Enable Server Result Cache, Oracle Wire Protocol 432
Enable SQLDescribeParam
Greenplum Wire Protocol 283
MySQL Wire Protocol 360

Oracle 722
Oracle Wire Protocol 432
PostgreSQL Wire Protocol 503
Enable Static Cursors for Long Data
Oracle 723
Oracle Wire Protocol 433
Enable Timestamp with Timezone
Oracle 724
Oracle Wire Protocol 434
encryption
data 103
SSL 104
Encryption Method
DB2 Wire Protocol 227
MySQL Wire Protocol 360
Oracle Wire Protocol 434
PostgreSQL Wire Protocol 504
SQL Server Native Wire Protocol 576
Sybase Wire Protocol 655
environment-specific information 32, 59
escape sequences 1040
example
bulk load sample application 127
character set conversions 128
creating a connection pool 110
Extended Column Metadata
Greenplum Wire Protocol 284
PostgreSQL Wire Protocol 505
extended connection failover 89
Extension Case, dBASE 878
Extension List, Text 975
extensions to SQL standards 1037
external overflow file 129

F

Failover Granularity
DB2 Wire Protocol 228
Greenplum Wire Protocol 285
MySQL Wire Protocol 361
Oracle Wire Protocol 435
PostgreSQL Wire Protocol 506
SQL Server Native Wire Protocol 577
Sybase Wire Protocol 655
Failover Mode
DB2 Wire Protocol 229
Greenplum Wire Protocol 286
MySQL Wire Protocol 362
Oracle Wire Protocol 436
PostgreSQL Wire Protocol 507
SQL Server Native Wire Protocol 578
Sybase Wire Protocol 656
Failover Preconnect
DB2 Wire Protocol 230
Greenplum Wire Protocol 286
MySQL Wire Protocol 363
Oracle Wire Protocol 437
PostgreSQL Wire Protocol 507
SQL Server Native Wire Protocol 579
Sybase Wire Protocol 657
FailoverNetworkAddress, Sybase Wire Protocol 659
Fetch Array Size, Sybase Wire Protocol 658
Fetch TSWTZ as Timestamp
Greenplum Wire Protocol 288
Oracle Wire Protocol 438
PostgreSQL Wire Protocol 509
SQL Server Classic Wire Protocol 772
SQL Server Native Wire Protocol 580
Fetch TWFS as Time
Greenplum Wire Protocol 288
PostgreSQL Wire Protocol 509
SQL Server Classic Wire Protocol 773
SQL Server Native Wire Protocol 580
FetchRefCursors
Greenplum Wire Protocol 287
PostgreSQL Wire Protocol 508
file data sources 167
File DSN 167
File Open Cache
Btrieve 847
dBASE 879
Paradox 940
Text 976
Flush Every Change, XML 1020
formats, for text files 958

FoxPro
configuring DBC data source 868
SQL statements 895

G

Get DB List From Informix, Informix 918
grammar, tokens used in SQL 1037
Grant Execute to 231
Greenplum Wire Protocol driver
connection failover 305
connection string attributes 277
connections supported 306
data source
configuring 264
connecting via connection string 274
connecting via logon dialog box 275
data types 300
driver requirements 263
isolation levels 305
locking levels 305
ODBC conformance 306
performance considerations 299
persisting result set as XML 305
statements supported 306
stored procedures 302
Unicode support 302
XML data type 302
GSS Client Library
DB2 Wire Protocol 232
Oracle Wire Protocol 438
SQL Server Native Wire Protocol 581
Sybase Wire Protocol 658
GUID literal 1039
guidelines for primary and alternate servers 94

H

HA Failover server, defining 659
hex literals 1040
hierarchical-formatted XML file support 993
High Availability Disaster Recovery (HADR) 94
hints, using with XML driver 1028
Host
MySQL 364
Host Name
Greenplum Wire Protocol 289
Informix 918
Informix Wire Protocol 326
PostgreSQL Wire Protocol 510
SQL Server Native Wire Protocol 582
Host Name In Certificate
DB2 Wire Protocol 232
MySQL Wire Protocol 364
Oracle Wire Protocol 440
PostgreSQL Wire Protocol 511
SQL Server Native Wire Protocol 583
Sybase Wire Protocol 660
HP-UX
See UNIX and Linux

I

IANAAppCodePage
DB2 Wire Protocol 233
dBASE 880
Greenplum Wire Protocol 290
Informix 919
Informix Wire Protocol 327
MySQL Wire Protocol 366
Oracle 724
Oracle Wire Protocol 441
PostgreSQL Wire Protocol 512
SQL Server Classic Wire Protocol 774
SQL Server Native Wire Protocol 584
Sybase Wire Protocol 661

Teradata 814
Text 976
identifiers
symbol in regular 1038
delimited 1038
regular 1038
Include Files with Matching Extensions, Text 977
Informix driver
See also Informix Wire Protocol driver
connection string attributes 912
connections supported 928
data source
configuring 901
connecting via connection string 908
connecting via logon dialog box 910
data types 923
driver requirements 900
enabling MTS 925
isolation levels 927
locking levels 927
MTS support 925
ODBC conformance 928
performance considerations 922
persisting result set as XML 927
statements supported 928
Informix Wire Protocol driver
See also Informix driver
connection failover 334
connection string attributes 320
connections supported 335
data source
configuring 310
connecting via connection string 317
connecting via logon dialog box 319
data types 332
driver requirements 309
enabling MTS 334
isolation levels 335
locking levels 335
ODBC conformance 335
performance considerations 331
persisting result set as XML 334
SQL grammar supported 335
statements supported 335
Initialization String
Greenplum Wire Protocol 291
Oracle Wire Protocol 442
PostgreSQL Wire Protocol 513
SQL Server Native Wire Protocol 585
Sybase Wire Protocol 662
Insert statement, XML 1036
integer numbers, XML 1038
Integrated Security, Teradata 815
Interactive Client, MySQL Wire Protocol 367
Interfaces File, Sybase Wire Protocol 663
InterfacesFileServerName, Sybase Wire Protocol 675
International Sort
Btrieve 848
dBASE 880
Paradox 941
Text 978
International Sort Order, XML 1013
Ip Address, DB2 Wire Protocol 234
IP addresses 70
IPv4 70
IPv6 70
isolation levels
Btrieve 858
DB2 Wire Protocol 259
dBASE 897
Greenplum Wire Protocol 305
Informix 927
Informix Wire Protocol 335
MySQL Wire Protocol 382
Oracle Wire Protocol 478, 748
Paradox 955
PostgreSQL Wire Protocol 531
SQL Server Classic Wire Protocol 787
SQL Server Native Wire Protocol 604
Sybase Wire Protocol 693
Teradata 829
ivtestlib tool 136

J

joins, XML 1037

K

Kerberos authentication
  kinit command 102
  requirements
    DB2 177
    Oracle 386
    SQL Server 536
    Sybase 608
  Ticket Granting Ticket (TGT) 102
Key Password
  DB2 Wire Protocol 235
  MySQL Wire Protocol 367
  Oracle Wire Protocol 442
  PostgreSQL Wire Protocol 513
  Sybase Wire Protocol 664
Keystore
  DB2 Wire Protocol 235
  MySQL Wire Protocol 368
  Oracle Wire Protocol 443
  PostgreSQL Wire Protocol 514
  Sybase Wire Protocol 664
Keystore Password
  DB2 Wire Protocol 236
  MySQL Wire Protocol 368
  Oracle Wire Protocol 443
  PostgreSQL Wire Protocol 514
  Sybase Wire Protocol 665
keywords, reserved, XML 1041
kinit command, Kerberos authentication 102

L

Language
  SQL Server Classic Wire Protocol 775
  SQL Server Native Wire Protocol 585
  Sybase Wire Protocol 665
LDAP authentication method 811
Left Outer Joins, XML 1037
Linux
  See UNIX and Linux
literal tokens, XML 1040
literals
  character string 1039
  date and time 1040
  GUID 1039
  hex 1040
  ODBC time and date 1040
  time 1040
  timestamp 1040
Load Balance Timeout
  DB2 Wire Protocol 237
  Greenplum Wire Protocol 291
  MySQL Wire Protocol 369
  Oracle Wire Protocol 444
  PostgreSQL Wire Protocol 515
  SQL Server Native Wire Protocol 586
  Sybase Wire Protocol 666
Load Balancing
  DB2 Wire Protocol 237
  Greenplum Wire Protocol 292
  Informix Wire Protocol 327
  MySQL Wire Protocol 369
  Oracle 725
  Oracle Wire Protocol 444
  PostgreSQL Wire Protocol 515
  SQL Server Classic Wire Protocol 775
  SQL Server Native Wire Protocol 586
  Sybase Wire Protocol 666
load balancing, about 95
Local Timezone Offset
  Oracle 726
  Oracle Wire Protocol 445
Location Name, DB2 Wire Protocol 238

Location Names, XML 1013
location_name, XML 1022
location_name.Catalog Type Hint, XML 1022
location_name.Create Type, XML 1026
location_name.Initial Catalog, XML 1021
location_name.Password, XML 1023
location_name.Require Passwd, XML 1024
location_name.Resolve External, XML 1025
location_name.Row Hint, XML 1025
location_name.Scan Rows, XML 1023
location_name.Table Hint, XML 1026
location_name.User ID, XML 1027
location_name.Validate Schema, XML 1027
locations for XML driver 996
Lock Compatibility, dBASE 881
Lock Timeout
  Oracle 726
  Oracle Wire Protocol 446
locking levels
  Btrieve 858
  DB2 Wire Protocol 259
  dBASE 897
  Greenplum Wire Protocol 305
  Informix 927
  Informix Wire Protocol 335
  MySQL Wire Protocol 382
  Oracle 748
  Oracle Wire Protocol 478
  Paradox 930, 955
  PostgreSQL Wire Protocol 531
  SQL Server Classic Wire Protocol 787
  SQL Server Native Wire Protocol 604
  Sybase Wire Protocol 693
  Teradata 829
Locking, dBASE 882
Logging, XML 1014
Login Timeout
  DB2 Wire Protocol 239
  Greenplum Wire Protocol 293
  MySQL Wire Protocol 370
  Oracle Wire Protocol 446
  PostgreSQL Wire Protocol 516
  SQL Server Native Wire Protocol 587
  Sybase Wire Protocol 667
  Teradata 816
lost connections
  extended connection failover 89
  recovering work in progress 92
  select failover 92

M

Map Call Escape to Exec, Teradata 816
materialized views
  Oracle 746
  Oracle Wire protocol 474
Max Pool Size
  DB2 Wire Protocol 239
  Greenplum Wire Protocol 293
  MySQL Wire Protocol 371
  Oracle Wire Protocol 447
  PostgreSQL Wire Protocol 517
  SQL Server Native Wire Protocol 588
  Sybase Wire Protocol 668
Maximum Response Buffer Size, Teradata 817
Microsoft Data Island format for XML 993
Min Pool Size
  DB2 Wire Protocol 240
  Greenplum Wire Protocol 294
  MySQL Wire Protocol 371
  Oracle Wire Protocol 448
  PostgreSQL Wire Protocol 517
  SQL Server Native Wire Protocol 588
  Sybase Wire Protocol 668
MTS support
  Informix 925
  Informix Wire Protocol 334
  Oracle 744
  Oracle Wire Protocol 473
  Sybase Wire Protocol 689
MySQL Wire Protocol driver
  connection failover 382
  connection string attributes 353
  connections supported 383

data source
configuring 338
connecting via connection string 350
connecting via logon dialog box 351
data types 380
driver requirements 338
isolation levels 382
locking levels 382
ODBC conformance 383
performance considerations 352
persisting result set as XML 382
SQL grammar supported 383
statements supported 383
Unicode support 381
wait timeout 367

N

Network Address, Sybase Wire Protocol 669
Network Directory, Paradox 942
Network, SQL Server Classic Wire Protocol 776
numbers
integer, XML 1038
real, XML 1039

O

ODBC conformance
Btrieve 858
DB2 Wire Protocol 260
dBASE driver 898
Greenplum Wire Protocol 306
Informix 928
Informix Wire Protocol 335
MySQL Wire Protocol 383
Oracle 749
Oracle Wire Protocol 478
Paradox 955
PostgreSQL Wire Protocol 532
SQL Server Classic Wire Protocol 788
SQL Server Native Wire Protocol 605
Sybase Wire Protocol 693
Teradata 829
Text 990
XML 1035
ODBC specification 58
ODBC, time and date literals, XML 1040
OEM to ANSI translation 79, 159
Open Database Connectivity
See ODBC specification
Optimize Long Performance, Oracle 727
OptimizePrepare, Sybase Wire Protocol 671
Oracle driver
See also Oracle Wire Protocol driver
authentication 706
connection failover 748
connection string attributes 713
connections supported 749
data encryption 706
data source
configuring 699
connecting via connection string 710
connecting via logon dialog box 711
data types 737
driver requirements 695
isolation levels 748
locking levels 748
materialized views 746
MTS support 744
ODBC conformance 749
performance considerations 733
persisting result set as XML 748
statements supported 749
stored procedures 746
unexpected characters 742
Unicode support 740
XML data type 738
Oracle Real Application Clusters (RAC) 474, 745
Oracle Wire Protocol driver
See also Oracle driver
authentication 395
bulk load operations, limitations 477

connection failover 477
connection pool
creating 110
dead connections 112
connection string attributes 416
connections supported 479
data encryption 395
data source
configuring 387
connecting via connection string 412
connecting via logon dialog box 413
data types 466
driver requirements 385
isolation levels 478
Kerberos authentication 386
locking levels 478
materialized views 474
MTS support 473
ODBC conformance 478
performance considerations 461
persisting result set as XML 477
statements supported 479
stored procedures 475
unexpected characters 470
Unicode support 470
XML data type 468
Outer Joins, XML 1037
overflow files for bulk load operations 129

P

Pack statement, using with dBASE driver 894
Package Collection, DB2 Wire Protocol 241
Package Name Prefix, DB2 Wire Protocol 241
Package Owner, DB2 Wire Protocol 242
Packet Size
SQL Server Native Wire Protocol 589
Packet Size, Sybase Wire Protocol 670
Paradox driver
Alter Table statement 945
connection string attributes 937
connections supported 955
Create Index Primary statement 952
Create Index statement 952
Create Table statement 946
data source
configuring 931
connecting via connection string 935
data types 943
decrypting table 949
driver requirements 929
Drop Index statement 953
dropping columns 945
encrypting table 948
encryption 947
index files 949
isolation levels 955
locking levels 930, 955
ODBC conformance 955
passwords
about 947
removing 949
Select statement 944
statements supported 955
table access, multiuser 930
transactions 954
parameter markers, binding 72
Password
Btrieve 848
DB2 Wire Protocol 242
Greenplum Wire Protocol 295
Informix 920
Informix Wire Protocol 328
MySQL Wire Protocol 372
Oracle 728
Oracle Wire Protocol 448
Paradox 947
PostgreSQL Wire Protocol 518
removing 949
SQL Server Native Wire Protocol 590
Sybase Wire Protocol 671
Teradata 817
XML 1023
performance considerations
DB2 Wire Protocol 250
Greenplum Wire Protocol 299

Informix 922
Informix Wire Protocol 331
MySQL Wire Protocol 352
Oracle 733
Oracle Wire Protocol 461
PostgreSQL Wire Protocol 524
SQL Server Classic Wire Protocol 783
SQL Server Native Wire Protocol 597
Sybase Wire Protocol 680, 692
Performance Wizard
product requirements 49
starting 50
using 50
persisting a result set as XML
DB2 Wire Protocol 259
Greenplum Wire Protocol 305
Informix 927
Informix Wire Protocol 334
MySQL Wire Protocol 382
Oracle 748
Oracle Wire Protocol 477
PostgreSQL Wire Protocol 531
SQL Server Native Wire Protocol 603
Sybase Wire Protocol 693
XML 829, 1035
Pervasive.SQL
See Btrieve driver
Port Number
Greenplum Wire Protocol 295
Informix Wire Protocol 329
MySQL Wire Protocol 372
Oracle Wire Protocol 449
PostgreSQL Wire Protocol 518
SQL Server Native Wire Protocol 591
Teradata 817
PostgreSQL Wire Protocol driver
authentication 489
connection failover 531
connection string attributes 497
connections supported 532
data encryption 489
data source
configuring 482
connecting via connection string 494
connecting via logon dialog box 495
data types 526
driver requirements 481
isolation levels 531
locking levels 531
ODBC conformance 532
performance considerations 524
persisting result set as XML 531
statements supported 532
stored procedures 528
Unicode support 528
XML data type 527
Prepare Method, Sybase Wire Protocol 671
primary server, specifying
guidelines for 94
PrintOption, Teradata 818
Procedure Returns Results
Oracle 728
Oracle Wire Protocol 449
ProcedureWithPrintStmt, Teradata 818
ProcedureWithSQLSource, Teradata 818
Profile, Teradata 819
Program ID, DB2 Wire Protocol 243
Protocol Type, Informix 920
PWD, SQL Server Classic Wire Protocol 776

Q

Query Timeout
DB2 Wire Protocol 244
Greenplum Wire Protocol 295
MySQL Wire Protocol 373
Oracle Wire Protocol 450
PostgreSQL Wire Protocol 519
SQL Server Native Wire Protocol 591
Sybase Wire Protocol 672
QueryLog_On, SQL Server Classic Wire Protocol 777

QueryLogFile, SQL Server Classic Wire Protocol 777
QueryLogTime, SQL Server Classic Wire Protocol 778

R

Read Only, XML 1014
Real Application Clusters (RAC) 474, 745
real numbers, XML 1039
Realm, Teradata 819
refcursor, PostgreSQL Wire Protocol 508, 530
references, aliasing table, XML 1037
Regional, SQL Server Classic Wire Protocol 778
regular identifiers, XML 1038
removing connections from a connection pool 111
Report Codepage Conversion Errors
  DB2 Wire Protocol 244
  Greenplum Wire Protocol 296
  MySQL Wire Protocol 374
  Oracle 729
  Oracle Wire Protocol 451
  PostgreSQL Wire Protocol 520
  SQL Server Native Wire Protocol 592
  Sybase Wire Protocol 673
  Teradata 819
  XML 1015
Report Recycle Bin
  Oracle 730
  Oracle Wire Protocol 452
reserved keywords for XML driver 1041
row hints for XML documents 1028
Rowid pseudo-column
  Btrieve 855
  dBASE 889
Rows to Scan, Text 978

S

SAVEFILE, SQL Server Classic Wire Protocol 779
ScanRows, Text 978
schema
  deleting externally linked files 1020
  format supported by XML driver 993
schema mode for XML driver 1016
Schema Mode, XML 1016
Secure Socket Layer
  See SSL
Security Mechanism, Teradata 820
Security Parameter, Teradata 823
security, using 101
select failover 92
Select Method, Sybase Wire Protocol 674
Select statement
  dBASE 889
  Paradox 944
  Text 989
  XML 1036
Server List 730
Server Name
  Informix 920
  Informix Wire Protocol 329
  Oracle 731
  Oracle Wire Protocol 452
  Sybase Wire Protocol 675
Server Process Type, Oracle Wire Protocol 453
server result cache, Oracle Wire Protocol 432
SERVER, SQL Server Classic Wire Protocol 779
service class, Workload Manager (WLM) 178, 252
Service Name
  Informix 921
  Oracle Wire Protocol 454
Service Principal Name, Sybase Wire Protocol 675
Session Character Set, Teradata 823
Show Manufactured Schemas, XML 1017, 1018

Show Selectable Tables, Teradata 824
SID, Oracle Wire Protocol 455
Snapshot Isolation level
  SQL Server Native Wire Protocol 604
Snapshot Isolation level, SQL Server Classic Wire Protocol 788
Solaris
  See UNIX and Linux
SQL
  comments 1046
  extensions to ANSI SQL 92 1037
  grammar token definitions 1037
  operators and symbols 1041
  standards, extensions to 1037
  syntax for date and time literals 1040
SQL Server Classic Wire Protocol driver
  connection failover 787
  connection string attributes
    UNIX and Linux 765
    Windows 763
  connections supported 789
  data source
    configuring 752
    connecting via connection string 759
    connecting via logon dialog box 761
  data types 784
  driver requirements 751
  isolation levels 787
  locking levels 787
  ODBC conformance 788
  statements supported 789
  Unicode support 786
SQL Server Native Wire Protocol driver
  authentication 545
  connection failover 603
  connection pool
    creating 110
    dead connections 112
  connection string attributes 562
  connections supported 605
  data encryption 545
  data source
    configuring 537
    connecting via connection string 559
    connecting via logon dialog box 560
  data types 599
  driver requirements 535
  isolation levels 604
  Kerberos authentication 536
  locking levels 604
  ODBC conformance 605
  persisting result set as XML 603
  statements supported 605
  Unicode support 602
  XML data type 601
SQL standards, extensions to 1037
SSL
  client authentication 107
  encryption 104
  server authentication 106
standards, ODBC specification compliance 58
statements supported
  Btrieve 859
  DB2 Wire Protocol 260
  dBASE 898
  Greenplum Wire Protocol 306
  Informix 928
  Informix Wire Protocol 335
  MySQL Wire Protocol 383
  Oracle Wire Protocol 479, 749
  Paradox 955
  PostgreSQL Wire Protocol 532
  SQL Server Classic Wire Protocol 789
  SQL Server Native Wire Protocol 605
  Sybase Wire Protocol 694
  Teradata 829
  Text 990
  XML 1036
StatsLog_On, SQL Server Classic Wire Protocol 781
StatsLogFile, SQL Server Classic Wire Protocol 780
stored procedures
  DB2 Wire Protocol 258
  Oracle 746

Oracle Wire Protocol 475
Sybase Wire Protocol 671
stored results
Greenplum Wire Protocol 302
Oracle 746
Oracle Wire Protocol 475
PostgreSQL Wire Protocol 528
string literals, character, XML 1039
SupportLink 35
Sybase Wire Protocol driver
ansinull option 690
authentication 617
bulk load
configuring 691
transactions 691
connection failover 692
connection pool
creating 110
dead connections 112
connection string attributes 639
connections supported 694
data encryption 617
data source
configuring 609
connecting via connection string 635
connecting via logon dialog box 637
data types 684
driver requirements 607
isolation levels 693
Kerberos authentication 608
locking levels 693
MTS support 689
NULL values, interpreting 690
ODBC conformance 693
performance considerations 680, 692
persisting result set as XML 693
statements supported 694
stored procedures 671
unexpected characters 686
Unicode support 685
symbol in regular identifiers, #, XML 1038
symbols, SQL, XML 1041
system information file (.odbc.ini) 137
system requirements
See driver requirements

T

table hints for XML documents 1028
table references, aliasing, XML 1037
table structure, defining
Btrieve 849
Text 980
Text under UNIX 984
tabular formats supported for XML 992
Tcp Port, DB2 Wire Protocol 245
TDProfile, Teradata 819
TDRole, Teradata 811
TDUserName, Teradata 825
Technical Support, contacting 35
Teradata, driver for
connections supported 829
data source
configuring 792
connecting via connection string 802
data types 826
driver requirements 791
isolation levels 829
locking levels 829
ODBC conformance 829
persisting result set as XML 829
statements supported 829
Unicode support 828
Text driver
Alter Table statement 989
connection string attributes 967
connections supported 990
data source
configuring 959
connecting via connection string 965
data types 988
date masks 986

defining table structure
on UNIX 984
on Windows 980
driver requirements 957
formats 958
ODBC conformance 990
Select statement 989
statements supported 990
Ticket Granting Ticket (TGT), Kerberos authentication 102
Tightly Coupled Distributed Transactions, Sybase Wire Protocol 676
time literals, XML 1040
Timestamp Escape Mapping
Oracle 732
Oracle Wire Protocol 455
timestamp literals, XML 1040
TNSNames File, Oracle Wire Protocol 456
tokens, XML
literal 1040
used in SQL grammar 1037
tracing, ODBC 140
Transaction Error Behavior
Greenplum Wire Protocol 297
PostgreSQL Wire Protocol 520
transactions
Btrieve 835
bulk copy on Sybase 691
effect of record locks with dBASE 897
Paradox 954
translators 83, 159
Treat Binary Data as Character Data, MySQL Wire Protocol 374
Trim Blank From Index Name
Informix 921
Informix Wire Protocol 329
Truncate Time Type Fractions, Sybase Wire Protocol 676
Trusted_Connection, SQL Server Classic Wire Protocol 781
Truststore
DB2 Wire Protocol 246
MySQL Wire Protocol 375
Oracle Wire Protocol 457
PostgreSQL Wire Protocol 521
SQL Server Native Wire Protocol 593
Sybase Wire Protocol 677
Truststore Password
DB2 Wire Protocol 246
MySQL Wire Protocol 376
Oracle Wire Protocol 458
PostgreSQL Wire Protocol 522
SQL Server Native Wire Protocol 593
Sybase Wire Protocol 677
typographical conventions 30

U

Unicode support
DB2 Wire Protocol 255
Greenplum Wire Protocol 302
MySQL Wire Protocol 381
Oracle 740
Oracle Wire Protocol 470
PostgreSQL Wire Protocol 528
SQL Server Classic Wire Protocol 786
SQL Server Native Wire Protocol 602
Sybase Wire Protocol 685
Teradata 828
XML 1035
UNIX and Linux
Data Source Administrator 46, 138
data source configuration 44, 137
driver names 70
drivers
DB2 Wire Protocol 173
dBASE 861
Greenplum Wire Protocol 263
Informix 899
Informix Wire Protocol 309
MySQL Wire Protocol 337
Oracle 695
Oracle Wire Protocol 385
PostgreSQL Wire Protocol 481
SQL Server Classic Wire Protocol 751
SQL Server Native Wire Protocol 535

Sybase Wire Protocol 607
Teradata 791
Text 957
environment
DD_INSTALLDIR 135
ddtestlib tool 136
introduction 131
ivtestlib tool 136
library search path 132
ODBCINI 133
ODBCINST 134
system information file (.odbc.ini) 137
variables 131
system requirements 62
Update statement, XML 1036
Use Current Schema for Catalog Functions, DB2 Wire Protocol 247
Use Current Schema for SQLProcedures
Oracle 732
Oracle Wire Protocol 458
Use Default Login, Informix 922
Use Delimited Identifier, Informix Wire Protocol 330
Use Long Names, dBASE 883
Use Long Qualifiers
Btrieve 849
dBASE 883
Paradox 942
Text 979
Use NT Authentication, SQL Server Classic Wire Protocol 781
Use Snapshot Transactions
SQL Server Classic Wire Protocol 782
SQL Server Native Wire Protocol 594
used in SQL grammar, tokens, XML 1037
user authentication 102
User Name
DB2 Wire Protocol 248
Informix Wire Protocol 330
MySQL Wire Protocol 376
Oracle Wire Protocol 459
PostgreSQL Wire Protocol 522
SQL Server Native Wire Protocol 595
user-defined functions, PostgreSQL Wire Protocol 528

V

Validate Server Certificate
DB2 Wire Protocol 248
MySQL Wire Protocol 377
Oracle Wire Protocol 459
PostgreSQL Wire Protocol 523
SQL Server Native Wire Protocol 595
Sybase Wire Protocol 678
validating bulk load files 123
version string information, drivers 74

W

Windows
Data Source Administrator 42
data source configuration 42
driver names 61
drivers
Btrieve 833
DB2 Wire Protocol 173
dBASE 861
Greenplum Wire Protocol 263
Informix 899
Informix Wire Protocol 309
MySQL Wire Protocol 337
Oracle 695
Oracle Wire Protocol 385
Paradox 929
PostgreSQL Wire Protocol 481
SQL Server Classic Wire Protocol 751
SQL Server Native Wire Protocol 535
Sybase Wire Protocol 607
Teradata 791
Text 957
XML 991
system requirements 59

Wire Protocol Mode, Oracle Wire Protocol 460
With Hold Cursors, DB2 Wire Protocol 249
Workload Manager (WLM) service class 178, 252
Workstation ID
  SQL Server Classic Wire Protocol 783
  SQL Server Native Wire Protocol 596
  Sybase Wire Protocol 679

X

XA Open String Parameters, Sybase Wire Protocol 679
XA Protocol
  Informix Wire Protocol 334
  Oracle 744
  Oracle Wire Protocol 473
  Sybase Wire Protocol 689
XML data file
  DB2 Wire Protocol 259
  Informix 927
  Informix Wire Protocol 334
  MySQL Wire Protocol 382
  Oracle Wire Protocol 477
  PostgreSQL Wire Protocol 531
  SQL Server Native Wire Protocol 603
  Sybase Wire Protocol 693
  XML 1035
XML data type
  DB2 Wire Protocol 254
  Greenplum Wire Protocol 302
  Oracle 738
  Oracle Wire Protocol 468
  PostgreSQL Wire Protocol 527
  SQL Server Native Wire Protocol 601
XML Describe Type
  DB2 Wire Protocol 249
  Greenplum Wire Protocol 298
  PostgreSQL Wire Protocol 524
  SQL Server Native Wire Protocol 597
XML driver
  connection string attributes 1011, 1019
  connections supported 1036
  data source
    configuring 1000
    connecting via connection string 1008
    connecting via logon dialog box 1010
  data types 1031
  defining locations 996
  driver requirements 992
  file formats supported 991
  hierarchical file format support 993
  hints for tabular-formatted XML documents 1028
  ODBC conformance 1035
  persisting result set as XML 1035
  reserved keywords 1041
  schema mode 1016
  specifying table names in SQL statements 998
  SQL statements supported 991
  SQL supported 1036
  statements supported 1036
  tabular file formats supported 992
  Unicode support 1035
XQuery expressions, DB2 Wire Protocol 258