

# DataDirect®

## Connect ODBC™

**Reference**

© 2000 MERANT. All rights reserved. Printed in the U.S.A.

DataDirect, INTERSOLV, MicroFocus, Middleware, Net Express, PVCS, SequeLink, and TechGnosis are registered trademarks, and Client/Server MiddleWare, DataDirect Connect ADO, DataDirect Connect Integrator, DataDirect Connect JDBC, DataDirect Connect ODBC, DataDirect Connect OLE DB, DataDirect Connect Premium, DataDirect Reflector, DataDirect SequeLink Integrator, MERANT, PVCS Dimensions, MERANT, PVCS Metrics, PVCS Replicator, PVCS TeamLink, PVCS Tracker, PVCS TrackerLink, PVCS Version Manager, PVCS VM Server, and WebDBLink are trademarks of MERANT. All other trademarks are the property of their respective owners.

ACKNOWLEDGEMENT. PVCS® Dimensions™ is implemented using the ORACLE® Relational database management system. ORACLE is a registered trademark of Oracle Corporation, Redwood City, California.

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

Licensees may duplicate the software product user documentation contained on a CD-ROM, 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.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is MERANT, 9420 Key West Avenue, Rockville, Maryland 20850. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

MERANT  
9420 Key West Avenue  
Rockville, Maryland 20850

# Table of Contents

<b>List of Tables</b> .....	<b>17</b>
<b>Preface</b> .....	<b>21</b>
What Is DataDirect Connect ODBC? .....	21
Using this Book .....	22
Conventions Used in This Book. ....	24
Typographical Conventions. ....	24
Environment-Specific Information .....	25
Other Connect ODBC Documentation .....	26
Ordering Printed Books. ....	27
Contacting Technical Support. ....	29
<b>1 Introduction</b> .....	<b>31</b>
About DataDirect Connect ODBC Drivers .....	31
Support for Multiple Environments .....	32
Installing the ODBC Drivers. ....	32
Environment-Specific Information .....	33
For Windows Users .....	33
For UNIX Users .....	34
Error Messages. ....	40
UNIX Error Handling .....	41
<b>2 Connect ODBC for Btrieve (Pervasive.SQL)</b> .....	<b>43</b>
Driver Requirements .....	44
Managing Databases. ....	45
Transactions .....	45

Configuring Data Sources . . . . .	46
Defining Table Structure . . . . .	52
Connecting to a Data Source Using a Connection String . . .	54
Data Types. . . . .	58
Indexes . . . . .	59
Column Names . . . . .	60
Select Statement. . . . .	60
Rowid Pseudo-Column . . . . .	60
Alter Table Statement . . . . .	61
Create and Drop Index Statements. . . . .	62
Create Index . . . . .	62
Drop Index . . . . .	63
Isolation and Lock Levels Supported. . . . .	63
ODBC Conformance Level . . . . .	63
Number of Connections and Statements Supported . . . . .	64
<b>3 Connect ODBC for DB2 . . . . .</b>	<b>65</b>
Driver Requirements . . . . .	65
Configuring Data Sources . . . . .	66
Connecting to a Data Source Using a Logon Dialog Box. . . .	77
Connecting to a Data Source Using a Connection String . . .	79
Data Types. . . . .	86
Unicode Support. . . . .	87
Default Unicode Mapping . . . . .	87
Connection Attributes for Unicode. . . . .	88
Persisting a Result Set as an XML Data File . . . . .	89
Stored Procedure Support . . . . .	91
Isolation and Lock Levels Supported. . . . .	91

ODBC Conformance Level . . . . .	91
Number of Connections and Statements Supported. . . . .	92
<b>4 Connect ODBC for dBASE . . . . .</b>	<b>93</b>
Driver Requirements . . . . .	93
Configuring Data Sources . . . . .	94
dBASE . . . . .	94
FoxPro 3.0 DBC. . . . .	100
Defining Index Attributes . . . . .	106
Defining Index Attributes on UNIX . . . . .	108
Connecting to a Data Source Using a Connection String . . . .	109
Data Types . . . . .	114
Column Names. . . . .	116
Select Statement . . . . .	116
Rowid Pseudo-Column . . . . .	117
Alter Table Statement . . . . .	118
Create and Drop Index Statements . . . . .	119
Create Index. . . . .	119
Drop Index . . . . .	121
Pack Statement . . . . .	122
SQL Statements for FoxPro 3.0 Database Containers . . . . .	123
Locking . . . . .	124
Levels of Database Locking. . . . .	124
Using Locks on Local Files . . . . .	125
Limit on Number of Locks . . . . .	125
How Transactions Affect Record Locks. . . . .	125
Isolation and Lock Levels Supported . . . . .	126
ODBC Conformance Level . . . . .	126
Number of Connections and Statements Supported. . . . .	126

<b>5</b>	<b>Connect ODBC for Excel Workbook . . . . .</b>	<b>127</b>
	Driver Requirements . . . . .	127
	Using an Excel Database . . . . .	127
	Configuring Data Sources . . . . .	128
	Connecting to a Data Source Using a Connection String . . .	133
	Data Types. . . . .	136
	SQL Supported . . . . .	136
	Table and Column Names . . . . .	136
	ODBC Conformance Level . . . . .	137
	Number of Connections and Statements Supported . . . . .	137
<b>6</b>	<b>Connect ODBC for Informix . . . . .</b>	<b>139</b>
	Driver Requirements . . . . .	139
	Windows . . . . .	139
	UNIX (AIX, HP-UX, and Solaris) . . . . .	140
	Configuring Data Sources . . . . .	141
	Connecting to a Data Source Using a Logon Dialog Box. . . .	147
	Connecting to a Data Source Using a Connection String . . .	148
	Data Types. . . . .	152
	MTS Support. . . . .	155
	Persisting a Result Set as an XML Data File . . . . .	156
	Isolation and Lock Levels Supported. . . . .	157
	ODBC Conformance Level . . . . .	158
	Number of Connections and Statements Supported . . . . .	158
<b>7</b>	<b>Connect ODBC for Informix Dynamic Server. . . . .</b>	<b>159</b>
	Driver Requirements . . . . .	159
	Configuring Data Sources . . . . .	159

Connecting to a Data Source Using a Logon Dialog Box . . . .	164
Connecting to a Data Source Using a Connection String . . . .	165
MTS Support . . . . .	167
Data Types . . . . .	168
Persisting a Result Set as an XML Data File . . . . .	170
Isolation and Lock Levels Supported . . . . .	171
ODBC Conformance Level . . . . .	172
Number of Connections and Statements Supported. . . . .	172
<b>8 Connect ODBC for MySQL. . . . .</b>	<b>173</b>
Driver Requirements . . . . .	173
Configuring and Connecting to Data Sources. . . . .	173
Data Types . . . . .	175
ODBC Conformance Level. . . . .	176
Number of Connections and Statements Supported. . . . .	176
<b>9 Connect ODBC for Oracle . . . . .</b>	<b>177</b>
Driver Requirements . . . . .	177
Windows. . . . .	177
UNIX . . . . .	178
Configuring Data Sources . . . . .	181
Connecting to a Data Source Using a Logon Dialog Box . . . .	187
Connecting to a Data Source Using a Connection String . . . .	188
Data Types . . . . .	193
Oracle8 . . . . .	194
Unicode Support . . . . .	195
Default Unicode Mapping. . . . .	195
Connection Attributes for Unicode . . . . .	196
Persisting a Result Set as an XML Data File . . . . .	197

- MTS Support . . . . . 199
- Stored Procedure Results . . . . . 199
- Isolation and Lock Levels Supported. . . . . 200
- ODBC Conformance Level . . . . . 200
- Number of Connections and Statements Supported . . . . . 201
- 10 Connect ODBC for Paradox. . . . . 203**
  - Driver Requirements . . . . . 203
  - Multuser Access to Tables . . . . . 204
    - Locking . . . . . 204
  - Configuring Data Sources . . . . . 205
  - Connecting to a Data Source Using a Connection String . . . 210
  - Data Types. . . . . 214
  - Select Statement. . . . . 216
    - Column Names . . . . . 216
  - Alter Table Statement . . . . . 216
    - Dropping Columns . . . . . 217
  - Create Table Statement . . . . . 217
  - Password Protection. . . . . 218
    - Encrypting a Paradox Table . . . . . 219
    - Accessing an Encrypted Paradox Table . . . . . 220
    - Decrypting a Paradox Table . . . . . 220
    - Removing a Password from Paradox. . . . . 220
    - Removing All Passwords from Paradox. . . . . 221
  - Index Files . . . . . 221
    - Primary Index . . . . . 221
    - Non-Primary Index . . . . . 222



Create and Drop Index Statements . . . . .	223
Create Index Primary Statement. . . . .	223
Create Index Statement. . . . .	224
Drop Index Statement . . . . .	225
Transactions . . . . .	226
Isolation and Lock Levels Supported . . . . .	226
ODBC Conformance Level . . . . .	227
Number of Connections and Statements Supported. . . . .	227
<b>11 Connect ODBC for PROGRESS. . . . .</b>	<b>229</b>
Driver Requirements . . . . .	229
Configuring Data Sources . . . . .	230
Remote OID with Direct Access. . . . .	230
Remote OID with Database Access via Server . . . . .	237
Configuring the Progress Environment . . . . .	243
Setting System Variables . . . . .	244
Services Files . . . . .	246
Connecting to a Data Source Using a Logon Dialog Box . . . .	247
Connecting to a Data Source Using a Connection String . . . .	250
Data Types . . . . .	253
Isolation and Lock Levels Supported . . . . .	253
ODBC Conformance Level . . . . .	254
Number of Connections and Statements Supported. . . . .	254
<b>12 Connect ODBC for SQL Server . . . . .</b>	<b>255</b>
Driver Requirements . . . . .	255
Windows. . . . .	255
UNIX . . . . .	256
Configuring Data Sources . . . . .	256
Connecting to a Data Source Using a Logon Dialog Box . . . .	260

Connecting to a Data Source Using a Connection String . . .	262
Windows . . . . .	263
UNIX . . . . .	269
Unicode Support. . . . .	272
Data Types. . . . .	273
Isolation and Lock Levels Supported. . . . .	274
ODBC Conformance Level . . . . .	274
Number of Connections and Statements Supported . . . . .	275
<b>13 Connect ODBC for SQL Server 6 . . . . .</b>	<b>277</b>
Driver Requirements . . . . .	277
Configuring Data Sources . . . . .	278
Connecting to a Data Source Using a Logon Dialog Box. . . .	284
Connecting to a Data Source Using a Connection String . . .	286
Data Types. . . . .	290
Persisting a Result Set as an XML Data File . . . . .	291
Isolation and Lock Levels Supported. . . . .	292
ODBC Conformance Level . . . . .	293
Number of Connections and Statements Supported . . . . .	293
<b>14 Connect ODBC for SQLBase . . . . .</b>	<b>295</b>
Driver Requirements . . . . .	295
Configuring Data Sources . . . . .	296
Connecting to a Data Source Using a Logon Dialog Box. . . .	300
Connecting to a Data Source Using a Connection String . . .	301
Data Types. . . . .	303
Isolation and Lock Levels Supported. . . . .	304

ODBC Conformance Level . . . . .	305
Number of Connections and Statements Supported. . . . .	305
<b>15 Connect ODBC for Sybase ASE . . . . .</b>	<b>307</b>
Driver Requirements . . . . .	307
Configuring Data Sources . . . . .	307
Connecting to a Data Source Using a Logon Dialog Box . . . .	318
Connecting to a Data Source Using a Connection String . . . .	319
Data Types . . . . .	326
MTS Support . . . . .	327
Unicode Support . . . . .	327
Default Unicode Mapping. . . . .	328
Connection Attributes for Unicode . . . . .	328
Persisting a Result Set as an XML Data File . . . . .	330
Support for Query Timeout. . . . .	331
Isolation and Lock Levels Supported . . . . .	332
ODBC Conformance Level . . . . .	332
Number of Connections and Statements Supported. . . . .	332
<b>16 Connect ODBC for Text . . . . .</b>	<b>333</b>
Driver Requirements . . . . .	333
Formats for Text Files. . . . .	334
Configuring Data Sources . . . . .	335
Defining Table Structure . . . . .	342
Defining Table Structure on UNIX Platforms . . . . .	346
Example of QETXT.INI . . . . .	348
Date Masks . . . . .	349
Connecting to a Data Source Using a Connection String . . . .	351

Data Types . . . . .	356
Select Statement . . . . .	357
Alter Table Statement . . . . .	357
ODBC Conformance Level . . . . .	358
Number of Connections and Statements Supported . . . . .	358
<b>17 Connect ODBC for XML . . . . .</b>	<b>359</b>
Driver Requirements . . . . .	360
Terminology . . . . .	360
Supported Formats for XML Files . . . . .	361
Configuring Data Sources . . . . .	362
Connecting to a Data Source Using a Logon Dialog Box . . . . .	367
Connecting to a Data Source Using a Connection String . . . . .	368
Using Hints . . . . .	372
Column Mode Identifier . . . . .	374
Data Types . . . . .	375
Persisting a Result Set as an XML Data File . . . . .	379
ODBC Conformance Level . . . . .	380
Number of Connections and Statements Supported . . . . .	381
SQL Supported . . . . .	381
SQL Statements . . . . .	381
Extensions to SQL Standards . . . . .	382
Grammar Token Definitions . . . . .	382
<b>A SQL for Flat-File Drivers . . . . .</b>	<b>393</b>
Select Statement . . . . .	393
Select Clause . . . . .	394
From Clause . . . . .	395
Where Clause . . . . .	396
Group By Clause . . . . .	397

Having Clause . . . . .	397
Union Operator . . . . .	398
Order By Clause . . . . .	399
For Update Clause . . . . .	399
SQL Expressions . . . . .	400
Create and Drop Table Statements . . . . .	411
Create Table . . . . .	411
Drop Table . . . . .	412
Insert Statement . . . . .	413
Update Statement . . . . .	415
Delete Statement . . . . .	416
Reserved Keywords . . . . .	417
<b>B Using Indexes . . . . .</b>	<b>419</b>
Introduction . . . . .	419
Improving Record Selection Performance . . . . .	421
Indexing Multiple Fields . . . . .	421
Deciding Which Indexes to Create . . . . .	423
Improving Join Performance . . . . .	425
<b>C ODBC API and Scalar Functions . . . . .</b>	<b>427</b>
API Functions . . . . .	427
Scalar Functions . . . . .	430
String Functions . . . . .	430
Numeric Functions . . . . .	433
Date and Time Functions . . . . .	435
System Functions . . . . .	437

<b>D</b>	<b>Locking and Isolation Levels</b>	<b>439</b>
	Locking	439
	Isolation Levels	440
	Locking Modes and Levels	443
<b>E</b>	<b>Threading</b>	<b>445</b>
<b>F</b>	<b>Performance Design of ODBC Applications</b>	<b>449</b>
	Optimizing Performance	449
	Catalog Functions	450
	Catalog Functions Are Relatively Slow	451
	Passing Null Arguments	451
	SQLColumns	453
	Retrieving Data	454
	Retrieving Long Data	454
	Reducing the Size of Data Retrieved	455
	Using Bound Columns	456
	Using SQLExtendedFetch Instead of SQLFetch	457
	ODBC Function Selection	459
	Using SQLPrepare/SQLExecute and SQLExecDirect	459
	Using SQLPrepare and Multiple SQLExecute Calls	460
	Using the Cursor Library	461
	Design Options	462
	Managing Connections	462
	Committing Data	463
	Asynchronous Execution	464
	Updating Data	465
	Using Positional Updates and Deletes	465
	Using SQLSpecialColumns	466

<b>G</b>	<b>Microsoft Query '97</b> .....	<b>469</b>
	Creating a Flat-File Data Source for Use with Microsoft Query '97.....	469
	Using Microsoft Query '97 with Single-Connect Data Sources.....	470
<b>H</b>	<b>The UNIX Environments</b> .....	<b>473</b>
	The System Information File (.odbc.ini).....	473
	Sample Solaris System Information File.....	474
	Environment Variables.....	477
	Required Environment Variables.....	477
	Optional Environment Variables.....	477
	Using Double-Byte Character Sets.....	478
	The ivtestlib Tool.....	479
	Translators.....	480
<b>I</b>	<b>Values for AppCodePage Connection String Attribute</b> .....	<b>481</b>
	<b>Index</b> .....	<b>485</b>





# List of Tables

Table 2-1.	Btrieve Connection String Attributes .....	55
Table 2-2.	Btrieve Data Types .....	58
Table 3-1.	DB2 Connection String Attributes .....	80
Table 3-2.	DB2 Data Types .....	86
Table 4-1.	dBASE Connection String Attributes .....	110
Table 4-2.	dBASE Data Types .....	115
Table 4-3.	Additional FoxPro 3.0 Data Types .....	116
Table 4-4.	dBASE-Compatible Index Summary .....	121
Table 5-1.	Excel Connection String Attributes .....	134
Table 5-2.	Excel Data Types .....	136
Table 6-1.	Informix Connection String Attributes .....	149
Table 6-2.	Informix Data Types .....	153
Table 7-1.	Informix Dynamic Server Connection String Attributes .....	166
Table 7-2.	Informix Data Types .....	168
Table 8-1.	MySQL Connection String Attributes .....	174
Table 8-2.	MySQL Data Types .....	175
Table 9-1.	Oracle Connection String Attributes .....	189
Table 9-2.	Oracle Data Types .....	193
Table 9-3.	Oracle8 Data Types .....	194
Table 10-1.	Paradox Connection String Attributes .....	211
Table 10-2.	Paradox Data Types .....	215

Table 11-1.	PROGRESS Connection String Attributes . . . . .	251
Table 11-2.	PROGRESS Data Types . . . . .	253
Table 12-1.	SQL Server Connection String Attributes on Windows . . . . .	263
Table 12-2.	SQL Server Connection String Attributes on UNIX . . . . .	269
Table 12-3.	SQL Server Data Types . . . . .	273
Table 13-1.	SQL Server 6 Connection String Attributes . . . . .	287
Table 13-2.	SQL Server 6 Data Types . . . . .	290
Table 14-1.	SQLBase Connection String Attributes . . . . .	302
Table 14-2.	SQLBase Data Types . . . . .	303
Table 15-1.	Sybase ASE Connection String Attributes . . . . .	320
Table 15-2.	Sybase ASE Data Types . . . . .	326
Table 16-1.	Common Text File Formats . . . . .	334
Table 16-2.	Date Masks for Text Driver . . . . .	349
Table 16-3.	Date Mask Examples . . . . .	350
Table 16-4.	Text Connection String Attributes . . . . .	352
Table 16-5.	Text Data Types . . . . .	356
Table 17-1.	Common XML File Formats . . . . .	361
Table 17-2.	XML Connection String Attributes . . . . .	369
Table 17-3.	Data Islands Data Types . . . . .	376
Table 17-4.	ADO 2.5 Persisted Files Data Types . . . . .	377
Table 17-5.	DataDirect Data Types . . . . .	378
Table 17-6.	SQL Extensions . . . . .	382
Table 17-7.	Reserved Keywords . . . . .	387
Table A-1.	Aggregate Functions . . . . .	395
Table A-2.	Relational Operators . . . . .	403
Table A-3.	Operator Precedence . . . . .	405

Table A-4.	Functions that Return Character Strings . . . . .	406
Table A-5.	Functions that Return Numbers . . . . .	409
Table A-6.	Functions that Return Dates. . . . .	410
Table C-1.	Function Conformance for 2.x ODBC Applications. . . . .	428
Table C-2.	Function Conformance for 3.x ODBC Applications. . . . .	429
Table C-3.	Scalar String Functions . . . . .	431
Table C-4.	Scalar Numeric Functions . . . . .	433
Table C-5.	Scalar Time and Date Functions . . . . .	435
Table C-6.	Scalar System Functions . . . . .	437
Table D-1.	Isolation Levels and Data Consistency . . . . .	442
Table E-1.	Threading Information . . . . .	447
Table F-1.	Common ODBC System Performance Problems and Solutions . . . . .	449
Table I-1.	AppCodePage Values . . . . .	481



# Preface

This book is your reference to MERANT™ DataDirect® Connect ODBC™. The DataDirect Connect ODBC product consists of a number of database *drivers* that are compliant with the Open Database Connectivity (ODBC) specification.

---

## What Is DataDirect Connect ODBC?

DataDirect Connect ODBC drivers enable you to connect to a variety of relational and flat-file databases from these platforms:

### Windows

- Windows 9x
- Windows Me
- Windows NT
- Windows 2000

### UNIX

- AIX
- HP-UX aCC Enabled
- Sun Solaris
- Linux (Red Hat, Caldera, SuSE)

---

## Using this Book

The content of this book is based on the assumption that you are familiar with your operating system and its commands. It contains the following chapters:

- An introductory chapter ([Chapter 1 “Introduction” on page 31](#)) that explains the DataDirect Connect ODBC drivers and ODBC, discusses environment-specific subjects, and explains the error messages returned by the drivers.
- 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 on which the drivers run, 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 more driver-specific information.

This book also includes several appendixes that provide information on technical topics:

- [Appendix A, “SQL for Flat-File Drivers” on page 393](#) explains the SQL statements that you can use with Btrieve, dBASE, Excel, Paradox, and text files.
- [Appendix B, “Using Indexes” on page 419](#) provides general guidelines on how to improve performance when querying a database system.
- [Appendix C, “ODBC API and Scalar Functions” on page 427](#) lists the ODBC API functions that each driver supports. Any exceptions are listed in the appropriate driver chapter, under the section “ODBC Conformance Level.” This appendix also lists the ODBC scalar functions.

- [Appendix D, “Locking and Isolation Levels” on page 439](#) provides a general discussion of isolation levels and locking.
- [Appendix E, “Threading” on page 445](#) discusses how ODBC ensures thread safety.
- [Appendix F, “Performance Design of ODBC Applications” on page 449](#) provides guidelines for designing performance-oriented ODBC applications.
- [Appendix G, “Microsoft Query ‘97” on page 469](#) discusses how to use ODBC with Microsoft Query ‘97.
- [Appendix H, “The UNIX Environments” on page 473](#) explains the structure of the *system information file* (used in the UNIX environment), provides a sample system information file, and discusses UNIX environment variables.
- [Appendix I, “Values for AppCodePage Connection String Attribute” on page 481](#) provides the valid values for the AppCodePage connection string attribute. This attribute is valid only for Connect ODBC drivers that run on UNIX.

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

For the latest information about the specific DataDirect drivers available for your platform, see the README file in your software package, or refer to the MERANT World Wide Web page at:

<http://www.merant.com>

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

---

## Conventions Used in This Book

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

### Typographical Conventions

This book uses the following typographical conventions:

Convention	Explanation
<i>italics</i>	Introduces new terms with which you may not be familiar, and is used occasionally for emphasis.
<b>bold</b>	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click <b>Next</b> .
UPPERCASE	Indicates the name of a file. For operating environments that use case-sensitive file names, the correct capitalization is used in information specific to those environments.  Also indicates keys or key combinations that you can use. For example, press the ENTER key.
monospace	Indicates syntax examples, values that you specify, or results that you receive.
<i>monospaced italics</i>	Indicates names that are placeholders for values that you specify. For example, <i>filename</i> .
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.



Convention	Explanation
brackets [ ]	Indicates optional items. For example, in the following statement: <code>SELECT [DISTINCT]</code> , <code>DISTINCT</code> is an optional keyword.  Also indicates sections of the Windows Registry.
braces { }	Indicates that you must select one item. For example, <code>{yes   no}</code> means that you must specify either <code>yes</code> or <code>no</code> .
ellipsis . . .	Indicates that the immediately preceding item can be repeated any number of times in succession. An ellipsis following a closing bracket indicates that all information in that unit can be repeated.

## Environment-Specific Information

This book pertains to Connect ODBC drivers for use under the following 32-bit operating environments:

<b>Microsoft</b>	Windows 9x, Windows Me, Windows NT, and Windows 2000
<b>UNIX</b>	AIX, HP-UX, Solaris, and Linux

Unless otherwise noted, information in this book applies to all of the Windows environments. Wherever information is provided that 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.

This book shows dialog boxes that are specific to Windows NT. If you are using the drivers in another Windows environment, the dialog box that you see may differ slightly from the Windows NT version. If you are using a graphical user interface in the UNIX

environment, you will see a dialog box for login, but not for driver configuration.

---

## Other Connect ODBC Documentation

This Reference and the *Connect ODBC Installation Guide* are provided on your DataDirect CD in PDF format, which allows you to view the books online or print them. You can view the Connect ODBC online documentation using Adobe Acrobat Reader. The DataDirect CD includes Acrobat Reader 4.x with Search. Connect ODBC product documentation is also available on the MERANT Web site:

<http://www.merant.com/products/datadirect/download/docs/dochome.asp>



On Windows platforms, online help is provided through the Help button on the Driver Setup windows and through a general ODBC driver help file, installed in the Connect ODBC program group, that can be accessed separately.



On UNIX platforms, online help for driver configuration is provided through this manual in PDF format, viewed with Acrobat Reader version 3.0 or higher. By default, this online book is not installed when you install the Connect ODBC product. Optionally, you can install the Connect ODBC online books when you install the product. In this case, the books are installed in the following location: *install\_dir/books/odbcref/odbcref.pdf*.

---

## Ordering Printed Books

As part of your Connect ODBC license agreement, you may print and distribute as many copies of the Connect ODBC books as needed.

If you do not want to print each of these online books, you can order printed versions from MERANT. To order, please complete the following order form and fax your request to MERANT at (919) 461-4526.



---

## Contacting Technical Support

MERANT provides technical support for all registered users of Connect ODBC, including limited installation support, for the first 30 days. For support after that time, contact us using one of the following methods or purchase further support by enrolling in the SupportNet program. For more information about SupportNet, contact your sales representative.

The MERANT Web site provides the latest support information through SupportNet Online, our global service network that provides access to valuable tools and information. Our SupportNet users access information using the Web, automatic email notification, newsgroups, and regional user groups. SupportNet Online includes a knowledge base that allows you to search on keywords for technical bulletins and other information. You also can download product fixes for your DataDirect products.

### World Wide Web

<http://support.merant.com>

### E-Mail

USA, Canada, and Mexico	<a href="mailto:datadirect.answerline@merant.com">datadirect.answerline@merant.com</a>
Australia and New Zealand	<a href="mailto:australia.answerline@merant.com">australia.answerline@merant.com</a>
Japan	<a href="mailto:jpn.answerline@merant.co.jp">jpn.answerline@merant.co.jp</a>
All other countries	<a href="mailto:int.datadirect.answerline@merant.com">int.datadirect.answerline@merant.com</a>

### Local Telephone Support

Local phone numbers can be found at:

<http://support.merant.com/websupport/contact/supportnetanswerline.asp>

Live Answerline telephone support is available 24 hours a day, seven days a week.

### Fax and Mail Information

Fax US	1 919 461 4527
Fax International	+32 (0) 15 32 09 19
Mail	1500 Perimeter Park Drive, Suite 100, Morrisville, NC 27560 USA

When you contact us, please provide the following information:

- The **product serial number** located on the Product Registration Information card or on a product serial number card in your package. The number will be checked to verify your support eligibility. If you do not have a SupportNet contract, we will ask you to speak with a sales representative.
- Your **name and organization**. For a first-time call, you may be asked for full customer information, including location and contact details.
- The **version number** of your DataDirect product.
- The type and version of your **operating system**.
- Any **third-party software or other environment information** required to understand the problem.
- A **brief description of the problem**, including any error messages that you have received, **and the steps preceding the occurrence of the problem**. Depending on the complexity of the problem, you may be asked to submit an example so that we can recreate the problem.
- An assessment of the **severity level** of the problem.

# 1 Introduction

This chapter contains the following sections:

- About DataDirect Connect ODBC Drivers
- Environment-Specific Information
- Error Messages

---

## About DataDirect Connect ODBC Drivers

The *drivers* that make up MERANT DataDirect Connect ODBC are compliant with the Open Database Connectivity (ODBC) specification. ODBC is a specification for an application program interface (API) that enables applications to access multiple database management systems using Structured Query Language (SQL).

ODBC permits maximum interoperability—a single application can access many different database management systems. This enables an ODBC developer to develop, compile, and ship an application without targeting a specific type of data source. Users can then add the database drivers, which link the application to the database management systems of their choice.

DataDirect provides ODBC drivers for both relational and flat-file database systems. The flat-file drivers provide full SQL support; see [Appendix A, “SQL for Flat-File Drivers” on page 393](#) for details.

## Support for Multiple Environments

DataDirect provides ODBC-compliant database drivers for the following operating systems:

- Windows 9x, Windows Me, Windows NT, Windows 2000
- UNIX: Solaris for SPARC, HP-UX, AIX, and Linux

NOTE: Database drivers are continually being added to each operating environment. See the README file shipped with your DataDirect product for an up-to-date list of drivers and for current driver information.

[“Environment-Specific Information” on page 33](#) explains the environment-specific differences of which you should be aware when using the database drivers in your operating environment.

## Installing the ODBC Drivers

The DataDirect Connect ODBC drivers are installed by the Setup program for the product with which they are shipped. For instructions on running the Setup program, see the Installation Guide that accompanies the product.



---

# Environment-Specific Information

The following topics contain information specific to your operating environment, such as file names and versions supported.



## For Windows Users

The Connect ODBC drivers support the following Windows operating systems:

- Windows 95
- Windows 98
- Windows NT with Service Pack 4 and higher
- Windows 2000
- Windows Me

On Windows, the ODBC drivers are 32-bit drivers. All required network software supplied by your database system vendors must be 32-bit compliant.

### *Starting the ODBC Administrator*

The "Configuring Data Sources" section in each driver chapter instructs you to start the ODBC Administrator to configure data sources. To start the ODBC Administrator, double-click the ODBC Data Sources icon in the Control Panel.

### *Driver Names*

The prefix for all Connect ODBC driver file names is "IV." The file extension is .DLL. This indicates that they are dynamic link libraries. For example, the Oracle7 driver file name is IVOR7nn.DLL, where *nn* is the revision number of the driver.

See the README file shipped with your DataDirect product for the file name of each driver.

## ***Disk Space and Memory Requirements***

Disk space requirements are 25 MB of free disk space on the disk drive where Windows is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 8 MB of memory on Windows 9x and Windows Me or at least 16 MB of memory on Windows NT and Windows 2000. If your system is hosting a relational database system, additional memory may be required. Consult your relational database documentation to determine the exact memory requirements.



## **For UNIX Users**

The following UNIX operating systems are supported:

- AIX
- HP-UX
- Solaris
- Linux (Red Hat, Caldera, and SuSE)

### ***AIX***

The ODBC drivers for AIX are supported on AIX 4.2.1+ and AIX 4.3. They are not supported on AIX 4.2.0.

The AIX ODBC drivers are only compatible with C++ built applications. Two versions of the ODBC drivers for AIX are provided. The default AIX ODBC drivers are for use on AIX 4.3 and with reentrant C++ applications on AIX 4.2 (applications built using an IBM reentrant C++ compiler such as x1C\_r).

Non-reentrant AIX ODBC drivers for AIX 4.2 are provided in the "nothread" subdirectory created during the AIX ODBC driver installation.

These non-reentrant AIX ODBC drivers are for use only with non-reentrant C++ applications on AIX 4.2 (applications built using a regular non-reentrant IBM C++ compiler such as xlc).

NOTE: The "nothread" directory is not a complete ODBC installation. It contains only the files required to replace the reentrant counterparts.

To replace the default ODBC files with the non-reentrant versions, change to the ODBC installation directory and execute the following:

```
cd nothread
cp -R * ..
```

## ***HP-UX aCC***

The Connect ODBC drivers for HP-UX aCC are supported on HP-UX 10.20 or higher when running with HP-UX applications built using the HP aCC compiler version 1.12 or higher.

NOTE: The ODBC drivers for HP-UX aCC are supported on HP-UX 11.0 provided the application using the ODBC drivers was built on HP-UX 10 using aCC 1.12 or higher. The ODBC drivers for HP-UX aCC are not supported when running with native HP-UX 11 applications.

The ODBC drivers require certain runtime library patches. The patch numbers are listed in the READ.ME file for your product. HP-UX patches are publicly available from the HP Web site ([www.hp.com](http://www.hp.com)) or FTP site ([i3107ffs.external.hp.com](ftp://i3107ffs.external.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 any of the specified patches on an HP

site and receive a message that a patch has been superseded, download and install the replacement patch.

### ***HP-UX 11 aCC Only***

The ODBC drivers for HP-UX 11 aCC are supported on HP-UX 11.0 or higher when running with HP-UX applications built using the HP aCC compiler version 3.05 or higher.

The ODBC drivers require certain runtime library patches. The patch numbers are listed in the READ.ME file for your product. HP-UX patches are publicly available from the HP Web site ([www.hp.com](http://www.hp.com)) or FTP site ([i3107ffs.external.hp.com](ftp://i3107ffs.external.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.

### ***Solaris***

The Connect ODBC drivers for Solaris are supported on Solaris 7 and 8. The drivers use the Solaris native (kernel) threading model and are built with SPARCompiler C++ 4.2. The drivers are compatible with applications that are built using this threading model and compiler.

### ***Linux***

The following Linux distributions are supported:

- Red Hat Linux 6.2
- Caldera OpenLinux 2.3
- SuSE Linux 6.4

The Connect ODBC drivers for Linux are supported only on Intel x86 machines running Linux.

The Connect ODBC drivers on Linux are only compatible with applications built using the g++ GNU project C++ Compiler version egcs-2.91.66 and that use the Linux native pthread threading model (Linuxthreads).

## ***The System Information File (.odbc.ini)***

In the UNIX environment, there is no ODBC Administrator. To configure a data source, you must edit the system information file, a plain text file that is normally located in the user's \$HOME directory and is usually called *.odbc.ini*. This file is maintained using any text editor to define data source entries as described in the "Connecting to a Data Source Using a Connection String" section of each driver's chapter. A sample file (odbc.ini) is located in the driver installation directory.

[Appendix H, "The UNIX Environments" on page 473](#) explains the structure of the system information file, provides a sample file, and discusses UNIX environment variables.

## ***Driver Names***

The Connect ODBC drivers are ODBC API-compliant dynamic link libraries, referred to in UNIX as *shared objects*. The prefix for all ODBC driver file names on UNIX is "iv." On UNIX, the driver file names are lowercase and the extension is *.so* or *.sl*. This is the standard form for a shared object. For example, the Oracle driver file name is *ivor7nn.so*, where *nn* is the revision number of the driver.

NOTE: The convention in this book is to list the driver names in uppercase with the extension *.DLL*.

See the README file shipped with your DataDirect product for the file name of each driver.

## ***Setting the Library Path Environment Variable***

You must include the full path to the dynamic link libraries in the environment variable `LD_LIBRARY_PATH` (on Solaris and Linux), `LIBPATH` (on AIX), and `SHLIB_PATH` (on HP-UX). For example, if you install the ODBC drivers in the system directory `/opt/odbc`, then the fully qualified path for the ODBC Pack is `/opt/odbc/lib`. During installation, a shell startup script is created and stored in the `odbc` directory. This shell script sets up the `odbc` environment for you.

For C shell users, the shell startup script is called `odbc.csh`. This script can be sourced from a user's own `.login` script. For example:

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

For Bourne or Korn shell users, the shell startup script is called `odbc.sh`. This script can also be sourced from a user's own `.profile` script. For example:

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

If you do not include the path `/opt/odbc/lib` in the environment variable `LD_LIBRARY_PATH` (on Solaris and Linux), `LIBPATH` (on AIX), and `SHLIB_PATH` (on HP-UX), then your applications are unable to load the ODBC drivers dynamically at runtime or to display error message text.

## ***Setting the Database Environment***

In addition to setting the environment variables required by your particular database client, you must also add the client's library directory to your shared library path. For example, to add the INFORMIX lib directory `/db/informix/lib` to the shared library path under Solaris and Linux, C shell users would enter:

```
setenv LD_LIBRARY_PATH /db/informix/lib:${LD_LIBRARY_PATH}
```

Bourne or Korn shell users would use:

```
LD_LIBRARY_PATH=/db/informix/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

See [“The ivtestlib Tool” on page 479](#) for information on diagnostic procedures using the ivtestlib tool.

## ***Disk Space and Memory Requirements***

Disk space requirements are 25 MB (40 MB on AIX) of free disk space on the disk where the UNIX system is installed.

Memory requirements vary, depending on the database driver. If you are using a flat-file database driver, you need at least 8 MB of memory. If your system will host a relational database system, additional memory will be required. Consult your relational database documentation to determine the exact memory requirements.

---

## Error Messages

Error messages can come from:

- An ODBC driver
- The database system
- The ODBC driver manager

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

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

*ODBC\_component* is the component in which the error occurred. For example, an error message from the MERANT SQL Server 6 driver would look like this:

```
[MERANT] [ODBC SQL Server 6 driver] Invalid precision specified.
```

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

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

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

With this type of message, *ODBC\_component* is the component that received the error from the data store indicated. For example, you may receive the following message from an Oracle data store:

```
[MERANT] [ODBC Oracle driver] [Oracle] ORA-0919: specified length too long for CHAR column
```

If you receive this type of error, something is incorrect regarding the database system. Check your database system documentation for more information or consult your database administrator. In this example, you would check your Oracle documentation.



The driver manager is a DLL that establishes connections with drivers, submits requests to drivers, and returns results to applications. An error that occurs in the driver manager has the following format:

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

For example, an error from the Microsoft driver manager might look like this:

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

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



## UNIX Error Handling

UNIX error handling follows the X/Open XPG3 messaging catalog system. Localized error messages are stored in the subdirectory `locale/localized_territory_directory/LC_MESSAGES`, where `localized_territory_directory` depends on your language.

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

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

```
[MERANT] [ODBC 20101 driver] 30040
```



## 2 Connect ODBC for Btrieve (Pervasive.SQL)

Connect ODBC for Btrieve (the "Btrieve driver") supports the following versions of Btrieve files in the Windows environments:

- Btrieve version 6.x
- Pervasive.SQL 7.x
- Pervasive.SQL 2000

The driver executes SQL statements directly on Btrieve databases.

See "[Environment-Specific Information](#)" on page 33 for detailed information about the Windows environments supported by this driver.

See the README file shipped with your DataDirect 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:

<b>Database Versions</b>	<b>Client Names</b>
Pervasive.SQL 2000	Pervasive.SQL 2000 client software
Pervasive.SQL 7.0	Pervasive.SQL 7.0 client software
Btrieve 6.15 for Windows 9x	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

Before you attempt to access Btrieve files, you must incorporate existing Btrieve files into a Scalable SQL database. See [“Defining Table Structure” on page 52](#) for information on defining table structure.

**NOTE:** The Btrieve driver may experience problems if the Btrieve Microkernel Engine's communication buffer size is smaller than the Btrieve driver's Array Size attribute. You can increase the communication buffer size with the Pervasive Software Setup Utility. You can decrease the array size option when you configure a data source using the ODBC Btrieve Driver Setup dialog box, or when passing a connection string.

---

## 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 don't 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.

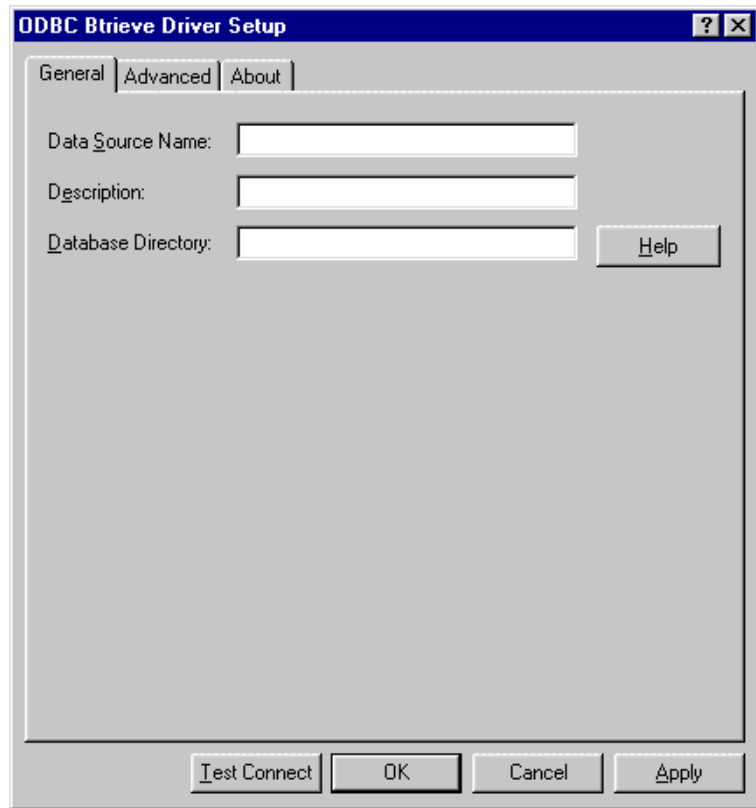
---

## Configuring Data Sources

Data sources are configured and modified through the ODBC Administrator. To configure a Btrieve data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Btrieve Driver Setup dialog box.

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



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.

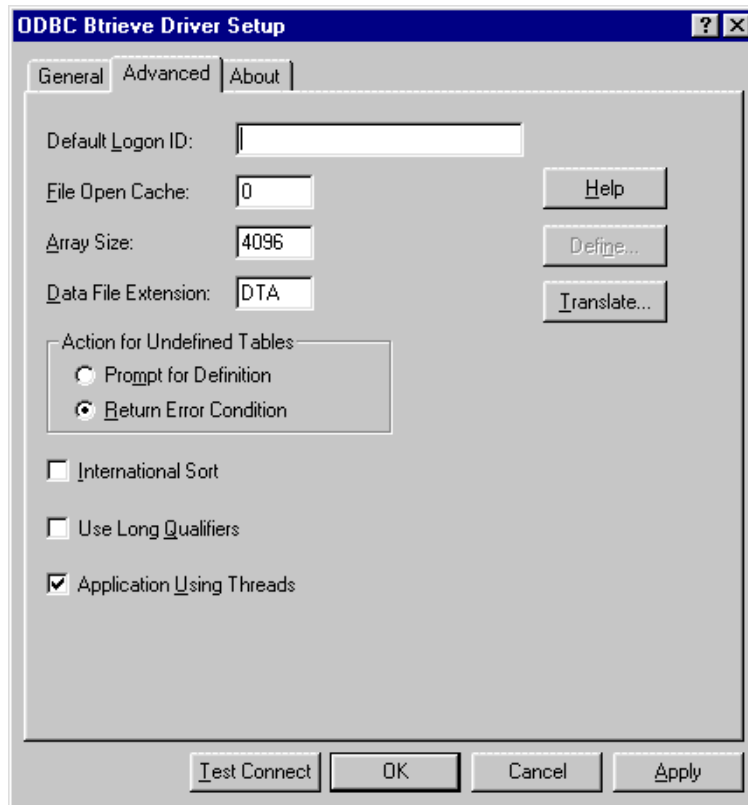
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this Btrieve data source configuration in the system information. Examples include "Accounting" or "Btrieve Files."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Btrieve files in C:\ACCOUNTS."

**Database Directory:** Type the full pathname of the directory that contains the Btrieve files and the data dictionary files (.DDF). Data dictionary files describe the structure of Btrieve data. If no directory is specified, the current working directory is used.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Default Logon ID:** Type the default logon ID used to connect to your Btrieve database. A logon ID is required only if security is enabled on your database. Your ODBC application



may override this value or you may override this value in a connection string.

**File Open Cache:** Type the numeric value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four 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 increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

**Array Size:** Type the number of bytes in the array. This attribute enables the driver to retrieve an array of records from the Btrieve engine and in most cases results in better performance for the application. The default value is 4,096 bytes, and the maximum is 65,535 bytes.

**Data File Extension:** Type a string of three or fewer characters that specifies the file extension to use for data files. The default value is DTA. This value is used for all Create Table statements. Sending a Create Table statement that uses an extension other than the one specified as the DataFileExtension value causes an error.

In other SQL statements, such as Select or Insert, you can specify an extension other than the DataFileExtension value. If you do not specify an extension value in these cases, the DataFileExtension value is used.

**Action for Undefined Tables:** Select one of these options to indicate whether the driver should prompt the user when it encounters a table for which it has no structure information. Select the Prompt for Definition option to prompt the user; select the Return Error Condition option (the default) to return an error.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

**Use Long Qualifiers:** Select this check box to specify whether the driver uses long path names. If you select the Use Long Qualifiers check box, path names can be up to 255 characters. If the check box is cleared (the default), the maximum path name length is 128 characters.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Define:** Click **Define** to define table structure. See ["Defining Table Structure"](#) on page 52 for step-by-step instructions.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC translators section of the system information. 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 and perform the translation.

- 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 improper environment or incorrect connection value, it will display 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**.

- 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.

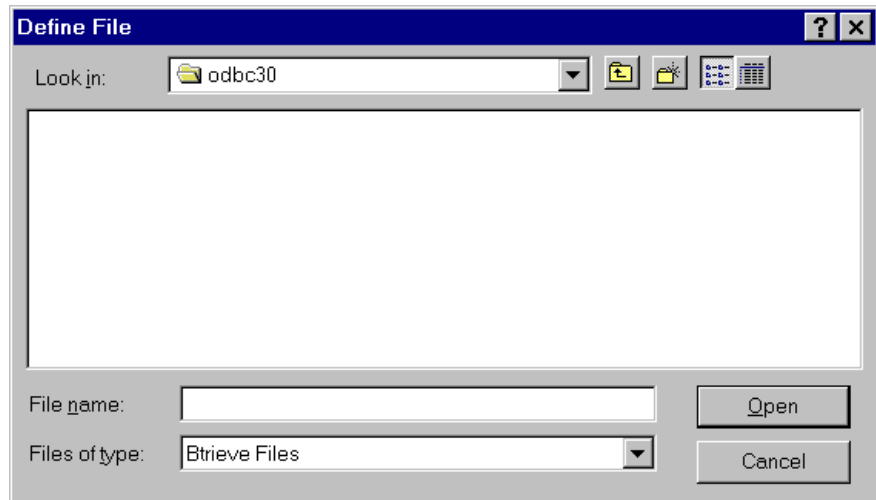
---

## Defining Table Structure

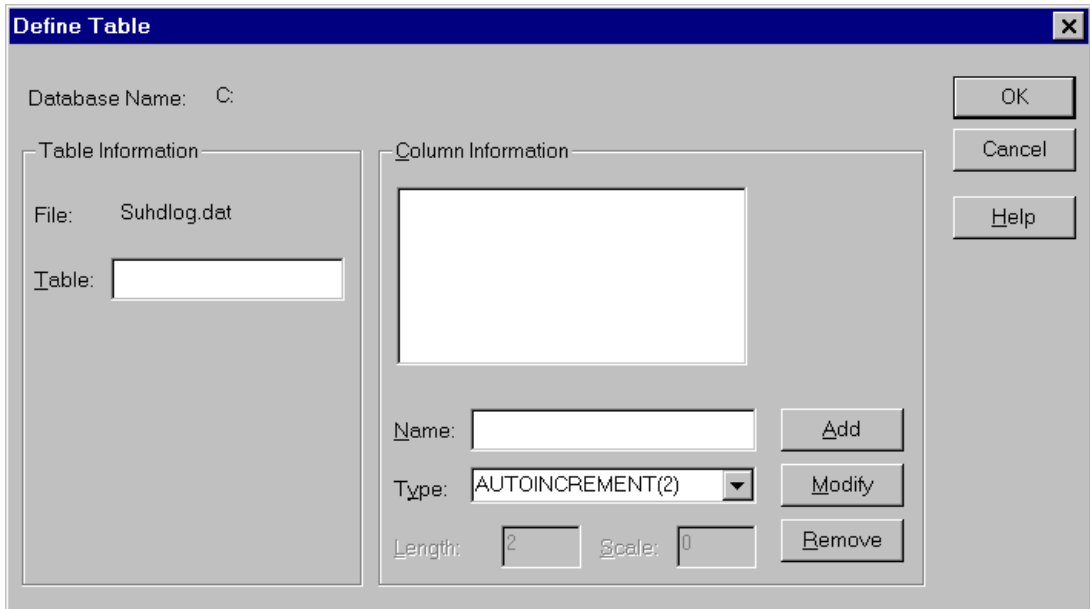
Because Btrieve does not store any column information in the data file, you may need to define its structure. Tables created by the DataDirect driver or by Scalable SQL will not require this. Utilities are also available from Pervasive that will perform this operation.

**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 Select the file you want to define and click **Open** to display the Define Table dialog box.



**Database Name:** The name of the Scalable SQL data dictionary directory that you selected in the Define File dialog box.

**File:** 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 up to 20 characters and cannot be the same as another defined table in the database. This field is required.

- 3 Enter 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.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Btrieve is:

```
DSN=BTRIEVE FILES;DB=J:\Btrvdata
```

[Table 2-1](#) lists the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in

the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 2-1. Btrieve Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe. When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
ArraySize (AS)	<p>An integer value that enables the driver to retrieve an array of records from the Btrieve engine and in most cases results in better performance for the application. The value of ArraySize is the number of bytes in the array.</p> <p>The initial default ArraySize is 4,096 bytes and the maximum is 65,535 bytes.</p>
Database (DB)	<p>The full pathname of the directory that contains the Btrieve files and the data dictionary files (.DDF). Data dictionary files describe the structure of Btrieve data. If no directory is specified, the current working directory is used.</p>
DataFileExtension (DFE)	<p>A string of three or fewer characters that specifies the file extension to use for data files. This value is used for all Create Table statements. If you execute a Create Table statement that uses an extension other than the one specified as the DataFileExtension value, an error occurs.</p> <p>In other SQL statements, such as Select or Insert, you can specify an extension other than the DataFileExtension value. If you do not specify an extension value in these cases, the DataFileExtension value is used.</p> <p>The initial default is DTA.</p>
DataSourceName (DSN)	<p>A string that identifies a Btrieve data source configuration in the system information. Examples include "Accounting" or "Btrieve Files."</p>

**Table 2-1. Btrieve Connection String Attributes** (cont.)

Attribute	Description
DeferQuery Evaluation (DQ)	<p data-bbox="498 317 1222 404">DeferQueryEvaluation={0   1}. Determines when a query is evaluated—after all records are read or each time a record is fetched.</p> <p data-bbox="498 423 1222 736">When set to 0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working with this result set.</p>
FileOpenCache (FOC)	<p data-bbox="498 755 1222 1069">When set to 1 (the initial default), the driver evaluates the query each time another record is fetched and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflect the latest changes to the database; however, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.</p> <p data-bbox="498 1088 1222 1446">An integer value that determines the maximum number of used file handles to cache. For example, when FileOpenCache=4, and a user opens and closes four 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 using file open caching is increased 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. The initial default is 0, which means no file open caching.</p>



**Table 2-1. Btrieve Connection String Attributes (cont.)**

<b>Attribute</b>	<b>Description</b>
IntlSort (IS)	<p data-bbox="528 303 1270 407">IntlSort={0   1}. Determines the order in which records are retrieved when you issue a Select statement with an Order By clause.</p> <p data-bbox="528 416 1270 546">When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p data-bbox="528 555 1270 746">When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>
LogonID (UID)	<p data-bbox="528 755 1270 894">The default logon ID used to connect to your Btrieve database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.</p>
Password (PWD)	<p data-bbox="528 902 1270 972">The password that you must enter if your Scalable SQL data dictionary files have security restrictions set.</p>
UndefinedTable (UT)	<p data-bbox="528 980 1270 1076">UndefinedTable={PROMPT   ERROR}. Determines whether the driver should prompt the user when it encounters a table for which it has no structure information.</p> <p data-bbox="528 1085 1270 1119">When set to PROMPT, the driver prompts the user.</p> <p data-bbox="528 1128 1270 1189">When set to ERROR (the initial default), the driver returns an error.</p>
UseLongQualifiers (ULQ)	<p data-bbox="528 1197 1270 1267">UseLongQualifiers={0   1}. Determines whether the driver uses long path names as table qualifiers.</p> <p data-bbox="528 1275 1270 1371">When set to 0 (the initial default), the driver does not use long path names (the maximum path name length is 128 characters).</p> <p data-bbox="528 1380 1270 1446">When set to 1, the driver uses long path names (the maximum path name length is 255 characters).</p>

## Data Types

Table 2-2 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 2-2. Btrieve Data Types**

<b>Btrieve</b>	<b>ODBC</b>
Autoincrement(2)	SQL_SMALLINT
Autoincrement(4)	SQL_INTEGER
Bfloat(4)	SQL_REAL
Bfloat(8)	SQL_DOUBLE
Bit	SQL_BIT
Blob	SQL_LONGVARGINARY
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
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

---

**Table 2-2. Btrieve Data Types (cont.)**


---

<b>Btrieve</b>	<b>ODBC</b>
Timestamp	SQL_TYPE_TIMESTAMP
Unsigned(1)	SQL_TINYINT
Unsigned(8)	SQL_BIGINT
Zstring	SQL_VARCHAR

---

## 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 Appendix A. See [Appendix A, “SQL for Flat-File Drivers” on page 393](#) for more information. 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. The Create Index statement is used to create indexes and the Drop Index statement is used to 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. See [Appendix D, “Locking and Isolation Levels” on page 439](#) for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions” on page 427](#) for a list of the API functions supported by the Btrieve driver. In addition, the following function is supported: SQLSetPos.

The Btrieve driver also supports backward and random fetching in `SQLExtendedFetch` and `SQLFetchScroll`. The driver supports the minimum SQL grammar with several core extensions.

---

## Number of Connections and Statements Supported

Btrieve files support a single connection and multiple statements per connection.



## 3 Connect ODBC for DB2

Connect ODBC for DB2 (the "DB2 driver") supports the following database systems in the following environments:

- DB2 Universal Database Versions 6 and 7 on Windows NT, Windows 2000, and UNIX
- DB2 for OS/390 Versions 5 and 6 (Connect Premium only)

The DB2 driver runs in the Windows and UNIX (not including Linux) environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the DB2 driver.

---

### Driver Requirements

The server requirement for all platforms is the same. The DB2 database must be installed as the Server Version (*not* the Local Version).

There is no client requirement for the DB2 driver.

---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.

Users must create the bind packages on every server to which they intend to connect with the driver. The driver will not work properly with any server that does not have the packages created. The UNIX version of the driver is provided with a program that creates the bind package. It is the equivalent of the Create Package button on the Bind tab of the DB2 driver setup. (See [Step 5](#) in this section.) To bind a package from a command shell, enter:

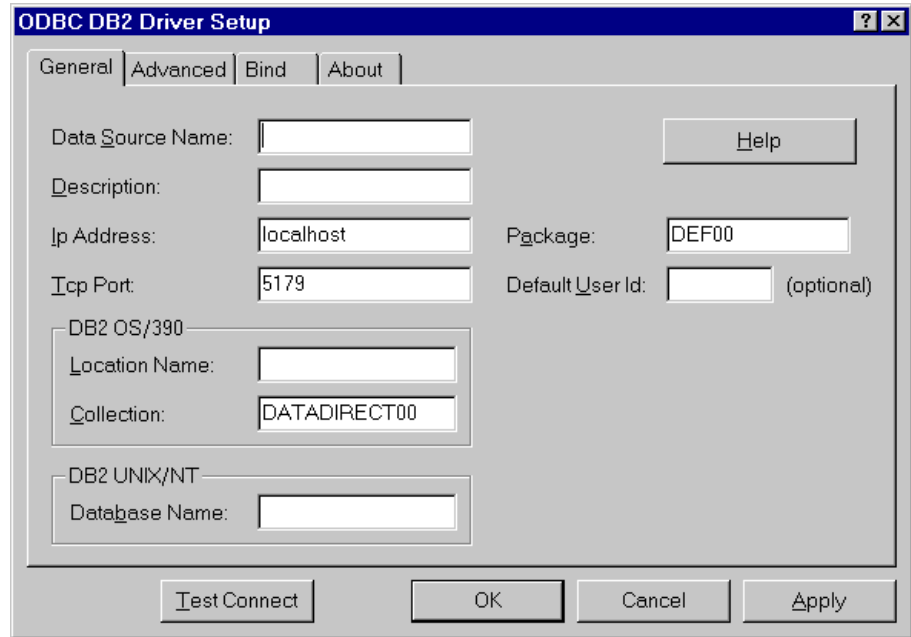
```
bind15 dsn
```

where *dsn* is the ODBC data source name. You are prompted for a user ID and password if they are not stored in the system information file.

### To configure a DB2 data source on Windows:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC DB2 Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the DB2 driver and click **Finish** to display the ODBC DB2 Driver Setup dialog box.



**NOTE:** The General and Bind tabs display only fields that are required for creating a data source. The fields on other tabs are optional, unless noted otherwise.

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

**Data Source Name:** Type a string that identifies this DB2 data source configuration in the system information. If you are creating a new data source definition, type a unique name of up to 32 characters. If you specify the name of an existing data source definition, the new settings will replace the existing ones.

**Description:** Type an optional descriptive comment for this data source definition. ODBC-related applications and development tools often display this description with the data source name when they display a list of data sources. If you want to include a description for this data source definition, type a comment of up to 64 characters.

**Ip Address:** 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.

**Tcp Port:** 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 (5179 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.

**Location Name:** This field is valid only if you are connecting to a DB2 database running on OS/390. Type the DB2 location name. Use the name defined during the local DB2 installation. NOTE: This field is disabled if the Database Name field is populated.

**Collection:** This field is valid only if you are connecting to a DB2 database running on OS/390. Type the name that identifies a group of packages. These packages include the Connect ODBC for DB2 driver packages. The default is DATADIRECT00. NOTE: This field is disabled if the Database Name field is populated.

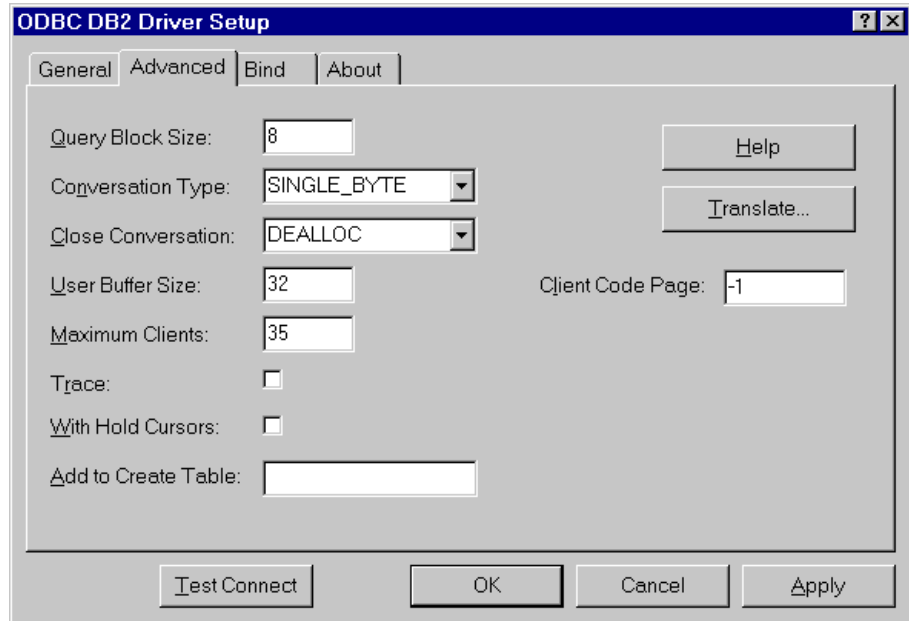
**Database Name:** This field is valid only if you are connecting to a DB2 database running on UNIX or NT. Type the name of the database to which you want to connect. NOTE: This field is disabled if the Location Name field is populated.

**Package:** Type the name of the package that the driver uses to process static and dynamic SQL for applications that use this data source definition. The default name is DEFxx, where xx is the version number.

**Default User ID:** Type the default user name used to connect to your DB2 database. Your ODBC application may override

this value or you may override it in the logon dialog box or connection string. This field is optional.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Query Block Size:** Type the number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user. The default is 8 rows.

**Conversation Type:** Select the type of system codes used by DB2. This value indicates the byte format of character data in the command area of a DB2 packet sent from the remote host. The possible values are:

- **Single Byte**—Used if the remote host uses a single-byte system code page (as specified in DSNZPARMS on DB2/MVS, for example).

- **Mixed Byte**—Used if the remote host is DB2 3.1 and DSNZPARMS specifies a double-byte code page where single-byte and mixed-byte values are not allowed.

**Close Conversation:** Select a value to determine when the DB2 driver closes an LU 6.2 conversation. The possible values are:

- **At Dealloc**—Conversation is closed when the client application terminates.
- **At Commit**—Conversation is closed after the client application executes a COMMIT statement.

Use the default value At Dealloc unless you are tuning the system for OLTP applications, or you want to prevent an application from using host resources when a client leaves it idle for an extended period of time.

**User Buffer Size:** Type the size (in kilobytes) of the bulk packet that the DB2 driver uses to download data from the host. Permitted values are 1 to 63. For most environments, the default value of 32 is sufficient; however, adjusting the value can optimize some client applications as follows:

- For client applications that frequently download large amounts of data, a large buffer size can improve response time.
- For client applications that perform brief online transactions, a small buffer will maximize memory on the machine where the DB2 driver is installed.

**Maximum Clients:** Type the maximum number of concurrent client sessions (maximum number of connections allowed) that the DB2 driver can carry. If the driver is used with an application server, select a value that will accommodate the number of users who will simultaneously access the host system through the DB2 driver. In a client configuration, use a small value to reduce the DB2 driver's memory requirements. The default is 10.

**Trace:** Select this check box to generate a trace file in the application's directory. The trace file name is packet.xxx, where xxx is an incremented number starting with 000.

**With Hold Cursors:** This option specifies the cursor behavior for the application used with this data source—either DB2 closes all open cursors (Delete cursors) after a commit or rollback or leaves them open (Preserve cursors). When this check box is selected, the cursor behavior is Preserve. Otherwise, the cursor behavior is Delete (the default).

If you are using the Static Bind Administrator and you want your package to use cursors WITH HOLD, you must select this check box. Note that any application using this package must use a data source with this option set.

When this option is enabled:

- The Static Bind Administrator automatically adds the WITH HOLD clause to queries that it puts in the application's database resource module (DBRM). The WITH HOLD clause prevents DB2 from automatically closing the cursor when the application executes a Commit statement.
- `SQLGetInfo()` returns `SQL_CB_PRESERVE` for `SQL_COMMIT_CURSOR_BEHAVIOR`.

When this option is not enabled, `SQLGetInfo()` returns `SQL_CB_DELETE`. For information about this function, refer to the Microsoft ODBC API.

**Add to Create Table:** Type a string that is automatically added to all Create Table statements. This field is primarily for users who need to add an "in database" clause.

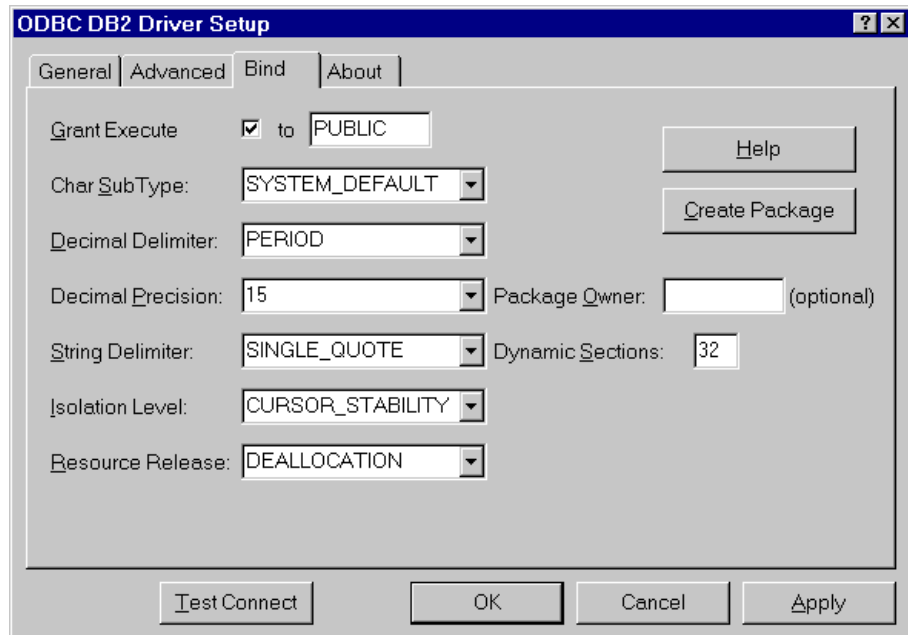
**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. DataDirect provides a translator named "OEM to ANSI" that translates

your data from the IBM PC character set to the ANSI character set.

Select a translator; then, click **OK** to close this dialog box and perform the translation.

- 5 Click the **Bind** tab to configure options for creating bind packages.

The Bind tab allows you to create the bind packages on the server that will be used by the driver. The tab also allows you to specify the behavior of the package. You must create the bind packages on every server to which you intend to connect with the driver. The driver will not work properly with any server that does not have the packages created.



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

**Grant Execute:** Select this check box to indicate whether or not to grant privileges on the package that you are creating.



The default value is grant execute privileges on the package to PUBLIC. You can also specify to whom to grant execute privileges.

**Char Sub Type:** Select a value to specify the options for storing ambiguous character data in the database. The possible values are:

- System Default—Specifies system default value of the DB2 location to which you are binding.
- SBCS—Specifies a single-byte character set.
- MBCS—Specifies a mixed-byte character set.
- DBCS—Specifies a double-byte character set.

**Decimal Delimiter:** Select a value to specify how the decimal point is represented. This is a required field if the DBRM file contains a decimal literal that does not match the host system's default value. The possible values are:

- Period—The decimal point is represented with a period, for example, 3.17. This is the default.
- Comma—The decimal point is represented with a comma, for example, 3,17.

**Decimal Precision:** Select a value to specify the number of places following the decimal point that will be calculated. Valid values are:

- 15—15 decimal places (the default)
- 31—31 decimal places

**String Delimiter:** Select a value to specify the type of quotation marks (single or double) used to represent constant string values that are referenced by the SQL in the DBRM. This option should match the option used to delimit the literal strings referenced by your embedded SQL. The default is single.

**Isolation Level:** Select the method by which locks are acquired and released by the system (see [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details). Valid values are:

- All—Prevents any other process from accessing data that your application has read or modified. All read or modified data is locked until the end of the transaction. Refer to "Serializable" in Appendix D.
- Change—Allows other processes to read from the database. Only modified data is locked until the end of the transaction.
- Cursor Stability (the default)—Allows other processes to change a row that your application has read if the cursor is not on the row you want to change. Prevents other processes from changing records that your application has changed until your program commits them or terminates. Prevents your program from reading a modified record that has not been committed by another process. Refer to "Read Committed" in Appendix D.
- No Commit—Allows your program to read modified records even if they have not been committed by another person. Refer to "Read Uncommitted" in Appendix D.
- Repeatable Read—Prevents other processes from changing records that are read or changed by your application (including phantom records) until your program commits them or terminates. Prevents the application from reading modified records that have not been committed by another process. If your program 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 program. Refer to "Repeatable Read" in Appendix D.

**Resource Release:** Select a value to specify the release of database resources. Two options are available:

- **Commit**—Releases database resources after a commit and provides a high level of concurrency.
- **Deallocation**—Releases database resources when the connection is terminated. This is the default.

**Dynamic Sections:** Type the number of statements that the DB2 driver package can prepare for a single user. The default is 32.

**Package Owner:** Type the AuthID assigned to the package. This DB2 AuthID must have authority to execute all the SQL in the package.

**Create Package:** Click to configure a package. Before you can use the DB2 driver, you must create a default DBRM for the link and bind it to DB2. The default DBRM creates a package that the DB2 driver uses to execute dynamic SQL statements from your application.

When you click the Create Package button, a logon dialog is displayed. Enter your user ID and password; then, click **Login**. A message is displayed if the package was not created successfully.

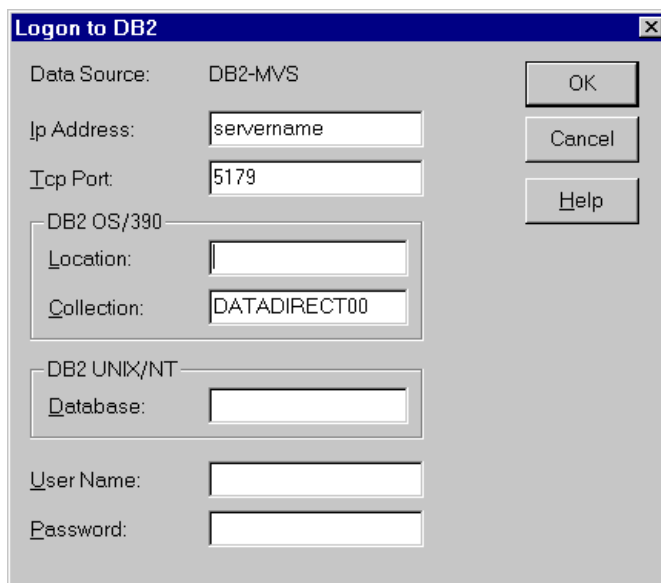
Each time that you bind a DBRM, the Bind Utility creates a log file (.LOG) in which it records all errors that occur during the bind. The .LOG file is stored locally in the same directory as your DBRM (.DBR). If no error has occurred during the binding of a particular DBRM, the log file does not exist.

- 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 is displayed; see [“Connecting to a Data Source Using a Logon Dialog Box” on page 77](#) 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 will display 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.

---

## Connecting to a Data Source 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. For DB2, the dialog box is as follows:



In this dialog box, perform the following steps:

- 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.
- 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 (5179 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 on OS/390, 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 In the Collection field, type the name that identifies a group of packages. These packages include the Connect ODBC for DB2 driver packages. The default is DATADIRECT00.
- 4 If you are running DB2 on Windows NT or UNIX, type the name of the database to which you want to connect in the Database field. Otherwise, skip to Step 5.
- 5 If required, type your authorization 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 system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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


An example of a connection string for DB2 is:

```
DSN=DB2MVS;LOC=TESTMVSDB2;UID=JOHN;PWD=XYZZY
```

[Table 3-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 3-1. DB2 Connection String Attributes**

Attribute	Description
AddStringTo CreateTable (ASCT)	A string that is automatically added to all Create Table statements. This field is primarily for users who need to add an "in database" clause.
AppCodePage (ACP) 	Valid values for this attribute are listed in <a href="#">Appendix I</a> . The code page that you specify must be the same as the code page used by your application.  The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order: <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).
CharSubType Type (CST)	CharSubTypeType={SYSTEM_DEFAULT   SBCS   MBCS   DBCS}. Determines which character set is specified.  When set to SYSTEM_DEFAULT, the system default value of the DB2 location to which you are binding is used.  When set to SBCS, a single-byte character set is used.  When set to MBCS, a mixed-byte character set is used.  When set to DBCS, a double-byte character set is used.
Close Conversation (CC)	CloseConversation={DEALLOC   AT_COMMIT}. Determines when the DB2 driver closes an LU 6.2 conversation.  When set to DEALLOC (the initial default), the conversation is closed when the client application terminates. Use this value unless you are tuning the system for OLTP applications, or you want to prevent an application from using host resources when a client leaves it idle for an extended period of time.  When set to AT_COMMIT, the conversation is closed after the client application executes a COMMIT statement.



**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
Collection (COL)	A name that identifies a group of packages. These packages include the Connect ODBC for DB2 driver packages. The default is DATADIRECT00. This attribute is valid only if you are connecting to DB2 on OS/390.
ConversationType (CT)	<p data-bbox="464 451 1283 581">ConversationType={SINGLE_BYTE   MIXED_BYTE}. Determines the type of system codes used by DB2. This value indicates the byte format of character data in the command area of a DB2 packet sent from the remote host.</p> <p data-bbox="464 590 1283 685">When set to SINGLE_BYTE, the remote host uses a single-byte system code page (as specified in DSNZPARMS on DB2/MVS, for example).</p> <p data-bbox="464 694 1283 789">When set to MIXED_BYTE, the remote host is DB2 3.1, and DSNZPARMS specifies a double-byte code page where single-byte and mixed-byte values are not allowed.</p>
Database (DB)	The name of the database to which you want to connect. This attribute is valid and required only if you are connecting to DB2 on Windows NT or UNIX.
DataSourceName (DSN)	A string that identifies a DB2 data source configuration in the system information. Examples include "Accounting" or "DB2-Serv1."
DecimalDelimiter (DD)	<p data-bbox="464 1006 1283 1137">DecimalDelimiter={COMMA   PERIOD}. Determines how the decimal point is represented. This is a required field if the DBRM file contains a decimal literal that does not match the host system's default value.</p> <p data-bbox="464 1145 1283 1223">When set to COMMA, the decimal point is represented with a comma, for example, 3,17.</p> <p data-bbox="464 1232 1283 1293">When set to PERIOD, the decimal point is represented with a period, for example, 3.17.</p>
DecimalPrecision (DP)	<p data-bbox="464 1302 1283 1362">DecimalPrecision={15   31}. Determines the number of places following the decimal point that will be calculated.</p> <p data-bbox="464 1371 1283 1449">When set to 15 (the initial default), 15 decimal places are calculated.</p> <p data-bbox="464 1458 1283 1484">When set to 31, 31 decimal places are calculated.</p>
DynamicSections (DS)	The number of statements that the DB2 driver package can prepare for a single user. The initial default is 32.

**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
GrantAuthid (GA)	A value that determines to whom execute privileges are granted. The default value is grant execute privileges on the package to PUBLIC.
GrantExecute (GE)	GrantExecute={0   1}. Indicates whether or not to grant privileges on the package that you are creating. When set to 0, privileges are not granted. When set to 1, privileges are granted.
IpAddress (IP)	The IP (Internet Protocol) address of the machine where the catalog tables are stored. Enter the address using the machine's numeric address (for example, 123.456.78.90) or type its address name. If you enter an address name, the driver must find this name (with the correct address assignment) in the HOSTS file on the workstation or in a DNS server.
IsolationLevel (IL)	IsolationLevel={ALL   CHANGE   CURSOR_STABILITY   NO_COMMIT   REPEATABLE_READ}. Specifies the method by which locks are acquired and released by the system (see <a href="#">Appendix D, "Locking and Isolation Levels" on page 439</a> for details). Valid values are: All—Prevents any other process from accessing data that your application has read or modified. All read or modified data is locked until the end of the transaction. Refer to "Serializable" in Appendix D. Change—Allows other processes to read from the database. Only modified data is locked until the end of the transaction. Cursor Stability—Allows other processes to change a row that your application read if the cursor is not on the row that you want to change. Prevents other processes from changing records that your application has changed until your program commits them or terminates. Prevents your program from reading a modified record that has not been committed by another process. Refer to "Read Committed" in Appendix D.

**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
IsolationLevel (IL) (cont.)	<p>No Commit—Allows your program to read modified records even if they have not been committed by another person. Refer to "Read Uncommitted" in Appendix D.</p> <p>Repeatable Read—Prevents other processes from changing records that are read or changed by your application (including phantom records) until your program commits them or terminates. Prevents the application from reading modified records that have not been committed by another process. If your program 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 program. Refer to "Repeatable Read" in Appendix D.</p>
Location (LOC)	A path that specifies the DB2 location name. Use the name defined during the local DB2 installation. This attribute is valid and required only if you are connecting to DB2 on OS/390.
LogonID (UID)	The default logon ID used to connect to your DB2 database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
MaximumClients (MC)	An integer that specifies the maximum number of concurrent client sessions that the DB2 driver can carry. If the driver is used with an application server, select a value that will accommodate the number of users who will simultaneously access the host system through the DB2 driver. In a client configuration, use a small value to reduce the DB2 driver's memory requirements. The default is 10.
Package (PCK)	The name of the package that the driver uses to process static and dynamic SQL for applications that use this data source definition. The default name is DEFxx, where xx is the version number.
PackageOwner (PO)	The AuthID assigned to the package. This DB2 AuthID must have authority to execute all the SQL in the package.
Password (PWD)	A password used to connect to your DB2 database.
QueryBlockSize (QBS)	<p>The number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user.</p> <p>The initial default is 8 rows.</p>

**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
ResourceRelease (RR)	<p>ResourceRelease={DEALLOCATION   COMMIT}. Specifies the release of database resources.</p> <p>When set to DEALLOCATION, database resources are released when the connection is terminated.</p> <p>When set to COMMIT, database resources are released after a commit and provide a high level of concurrency.</p>
StringDelimiter (SD)	<p>StringDelimiter={SINGLE_QUOTE   DOUBLE_QUOTE}. Specifies the type of quotation marks (single or double) used to represent constant string values that are referenced by the SQL in the DBRM. This option must match the option used to delimit the literal strings referenced by your embedded SQL.</p>
TcpPort (PORT)	<p>The port number that is assigned to the DB2 server on the machine where the catalog tables are stored. Specify this port's numeric address or its name (5179 is the default port address). If you specify a port name, the driver must find this name (with the correct port assignment) in the SERVICES file on the workstation.</p>
Trace (TR)	<p>Trace = {0   1}. Determines whether a trace file is created in the application's directory. When Trace=1, a trace file is created. The trace file name is packet.xxx, where xxx is an incremented number starting with 000.</p>
UserBufferSize (UBS)	<p>The size in kilobytes of the bulk packet that the DB2 driver uses to download data from the host. Valid values are 1 to 63. For most environments, the default value of 32 is sufficient; however, adjusting the value can optimize some client applications as follows:</p> <ul style="list-style-type: none"> <li data-bbox="428 1203 1246 1263">■ For client applications that frequently download large amounts of data, a large buffer size can improve response time.</li> <li data-bbox="428 1291 1246 1383">■ For client applications that perform brief online transactions, a small buffer will maximize memory on the machine where the DB2 driver is installed.</li> </ul>

**Table 3-1. DB2 Connection String Attributes** (cont.)

Attribute	Description
WithHold (WH)	<p data-bbox="464 314 1278 473">WithHold={0   1}. Specifies the cursor behavior for the application used with this data source—either DB2 closes all open cursors (Delete cursors) after a commit or rollback, or leaves them open (Preserve cursors). When set to 1, the cursor behavior is Preserve. When set to 0, the cursor behavior is Delete (the default).</p> <p data-bbox="464 487 1278 612">If you are using the Static Bind Administrator and you want your package to use cursors WITH HOLD, you must set this attribute to 1. Note that any application using this package must use a data source with this attribute set to 1.</p> <p data-bbox="464 626 649 661">When set to 1:</p> <ul data-bbox="464 678 1278 933" style="list-style-type: none"> <li data-bbox="464 678 1278 847">■ The Static Bind Administrator automatically adds the WITH HOLD clause to queries that it puts in the application's database resource module (DBRM). The WITH HOLD clause prevents DB2 from automatically closing the cursor when the application executes a Commit statement.</li> <li data-bbox="464 869 1049 933">■ SQLGetInfo( ) returns SQL_CB_PRESERVE for SQL_COMMIT_CURSOR_BEHAVIOR.</li> </ul> <p data-bbox="464 951 1278 1010">When set to 0, SQLGetInfo( ) returns SQL_CB_DELETE. For information about this function, refer to the Microsoft ODBC API.</p>

## Data Types

Table 3-2 shows how the DB2 data types map to the standard ODBC data types.

**Table 3-2. DB2 Data Types**

<b>DB2</b>	<b>ODBC</b>
Char	SQL_CHAR
Char() for Bit Data	SQL_BINARY
Clob*	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Dbclob*	SQL_LONGVARBINARY
Decimal	SQL_DECIMAL
Float	SQL_DOUBLE
Graphic	SQL_BINARY
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Long Varchar for Bit Data	SQL_LONGVARBINARY
Long Vargraphic	SQL_LONGVARBINARY
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR
Varchar() for Bit Data	SQL_VARBINARY
Vargraphic	SQL_VARBINARY

\*Only the first 32K of the Clob and DBClob data types are returned when fetching this data type from DB2 databases. Also, only 32K can be inserted and updated on DB2 databases.

---

# Unicode Support

The DB2 driver supports Unicode data types if the database was created with a multi-byte character set.

The driver maps the following DB2 data types to Unicode data types:

<b>DB2 Data Type</b>	<b>Mapped to. . .</b>
Char	SQL_WCHAR
Clob*	SQL_WLONGVARCHAR
Long Varchar	SQL_WLONGVARCHAR
Varchar	SQL_WVARCHAR

\*Only the first 32 KB of the Clob data type is returned when fetching this data type from DB2 databases. Also, only 32 KB can be inserted and updated on DB2 databases.

This driver supports the Unicode ODBC function calls, called W (Wide) calls (for example, SQLConnectW). These calls are used to accept Unicode datastreams.

## Default Unicode Mapping

The default Unicode mapping for an application's SQL\_C\_WCHAR variable is:

<b>Platform</b>	<b>Default Unicode Mapping</b>
Windows	UCS-2
AIX	UTF-8
HP-UX	UTF-8
Solaris	UTF-8
Linux	UTF-8

## Connection Attributes for Unicode

Two new connection attributes are available to support Unicode. These attributes determine how character data is converted and presented to an application and the database. The connection attributes are:

SQL_ATTR_APP_WCHAR_TYPE (1061)	Sets the SQL_C_WCHAR type for parameter and column binding to the desired unicode type, either SQL_DD_CP_UCS2 or SQL_DD_CP_UTF8. The default is the default Unicode mapping (see the previous section, <a href="#">“Default Unicode Mapping”</a> ).
SQL_ATTR_DBMS_CODE_PAGE (1062)	Sets the code page type of the database. The main purpose of this attribute is to set the code page type of a database for drivers that cannot determine the database’s code page. The default is SQL_DD_CP_ANSI.

Valid values for the two connection attributes are:

- SQL\_DD\_CP\_ANSI (ANSI code page)
- SQL\_DD\_CP\_UCS2 (UCS-2 code page)
- SQL\_DD\_CP\_UTF8 (UTF-8 code page)

You can set these connection attributes either before or after a connection is made. If the connection attributes are changed after a connection is established, all conversions use the new values.



The driver does not verify that the connection attributes are set only once per connection. Conversions are made based on the current application and database settings.

If the application does **not** set the `SQL_ATTR_DBMS_CODE_PAGE` attribute, the driver tries to determine the database's code page type; if the driver cannot determine the code page type, the driver sets the `SQL_ATTR_DBMS_CODE_PAGE` attribute to `SQL_DD_CP_ANSI`. If the application does set the `SQL_ATTR_DBMS_CODE_PAGE` attribute, the driver, even if it can determine the database's code page type, does not override the value of the attribute set by the application.

If a driver does not support Unicode, `SQLGetConnectAttr` and `SQLSetConnectAttr` return `HYC00`.

These new connection attributes and their valid values can be found in the file `qesqlx.h`, which is installed with the driver.

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on `STATIC` cursors. For example:

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

- 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 regarding Step 1:** 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.

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, SQL\_PERSIST\_AS\_XML. A client application must call SQLSetStmtAttr with this new attribute as an argument. See the following table for the definition of valid arguments for SQLSetStmtAttr.

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

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

Function Sequence Error

---

## Stored Procedure Support

The DB2 driver supports DB2 Remote Procedure Calls (RPCs) with the following restrictions:

- Multiple result sets are not returned; only the first result set is 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.

---

## Isolation and Lock Levels Supported

DB2 supports isolation levels 0 (read uncommitted), 1 (read committed), and 2 (repeatable read). It supports record-level locking. See [Appendix D, “Locking and Isolation Levels” on page 439](#) for details.

NOTE: An isolation level can be set only before connecting to a DB2 database.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions” on page 427](#) for a list of the API functions supported by the DB2 driver. In addition, the following X/Open functions are supported:

- SQLProcedures
- SQLProcedureColumns

The driver supports the minimum SQL grammar.

## Number of Connections and Statements Supported

The DB2 database system supports multiple connections and multiple statements per connection.

## 4 Connect ODBC for dBASE

Connect ODBC for dBASE (the "dBASE driver") supports the following file types in the Windows and UNIX environments:

<b>File Type</b>	<b>Operating Environments</b>
dBASE IV and V	Windows and UNIX
Clipper	Windows
FoxPro 2.5 and 2.6	Windows
FoxPro 3.0	Windows and UNIX
FoxPro 3.0 database container (DBC)	Windows

On Windows, FoxPro 6.0 is supported when using FoxPro 3.0 functionality. New FoxPro 6.0 functionality is not supported.

See [“Environment-Specific Information” on page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the dBASE driver.

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.

---

### Driver Requirements

There are no client requirements for the dBASE driver.

---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.



In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 4-1 on page 110](#). You must also edit this file to perform a translation. See [Appendix H, "The UNIX Environments" on page 473](#) for information about editing the file.

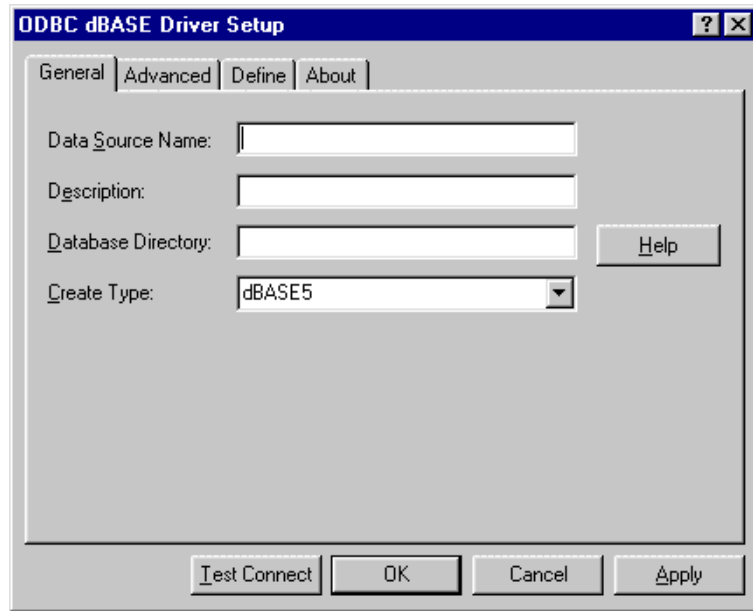
NOTE: To configure a data source for a FoxPro 3.0 database container (DBC), see ["FoxPro 3.0 DBC" on page 100](#).

### dBASE

To configure a dBASE data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC dBASE Driver Setup dialog box.

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



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

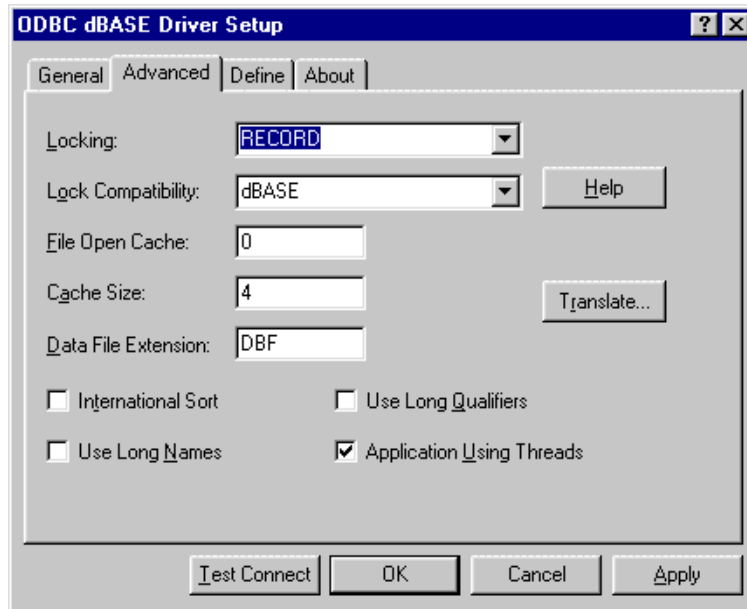
**Data Source Name:** Type a string that identifies this dBASE data source configuration in the system information. Examples include "Accounting" or "dBASE Files."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "dBASE files in C:\ACCOUNTS."

**Database Directory:** Type the path to the directory that contains the database files. If none is specified, the current working directory is used.

**Create Type:** Select the type of table or index to be created on a Create Table or Create Index statement. The default is dBASE V.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Locking:** Select the level of locking for the database file. FILE locks all of the records in the table. RECORD (the default) locks only the records affected by the statement. NONE offers the best performance but is intended only for single-user environments.

**Lock Compatibility:** Select the locking scheme the driver uses when locking records. Select Clipper, dBASE, Fox, Q+E, or Q+EVirtual. The default is dBASE. 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. These values determine locking support as follows:

- 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.

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 are accessing 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.

**File Open Cache:** Type an integer value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four 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 increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though

no one appears to have the file open. The default is 0, which means no file open caching.

**Cache Size:** Type the number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 will not be able to see updates made by other users until you run the Select statement again. The default is 4.

**Data File Extension:** Type the file extension to use for data files. The default setting is DBF. The setting cannot be greater than three characters, and it cannot be one the driver already uses, such as MDX or CDX. The Data File Extension setting 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 attribute. The DataFileExtension value is used when no extension is specified.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

**Use Long Names:** Select this check box to use long file names as table names. The maximum table name length is specific to

the environment in which you are running (for example, in Windows 9x, the maximum table name length is 128).

**Use Long Qualifiers:** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for path names is 128 characters.

**Applications Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 5 If you use index files that have different names than their corresponding data files and you have not defined this association, click the **Define** tab. See ["Defining Index Attributes" on page 106](#) 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 improper environment or incorrect connection value, it will display an appropriate error message. Click **OK**.

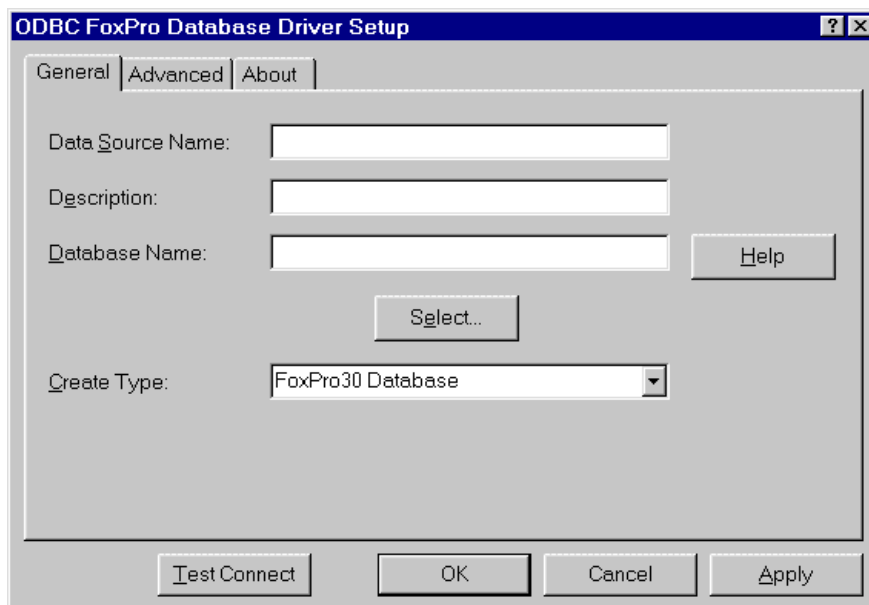
- 7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

## FoxPro 3.0 DBC

To configure a data source for FoxPro 3.0 database containers:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC FoxPro Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the FoxPro3.0 driver and click **Finish** to display the ODBC FoxPro Driver Setup dialog box.



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

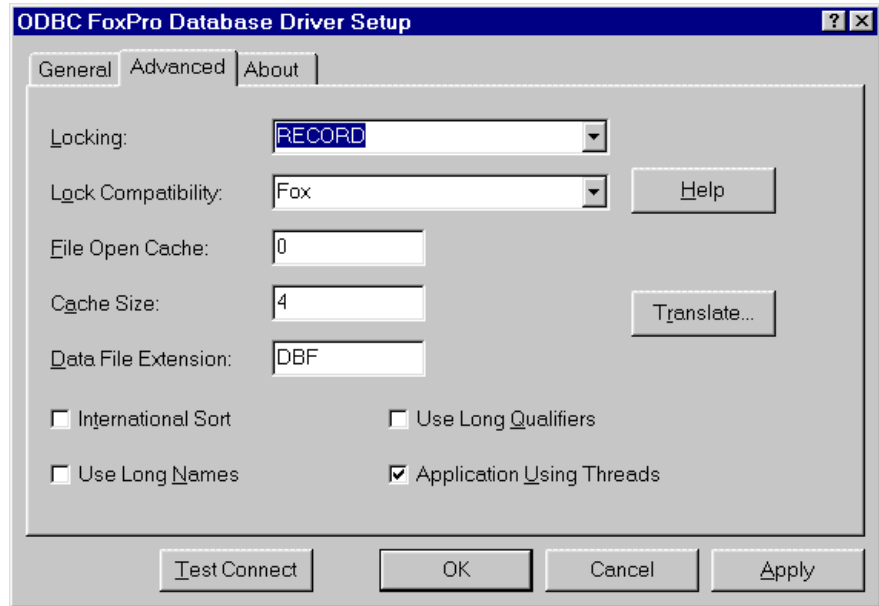
**Data Source Name:** Type a string that identifies this FoxPro data source configuration in the system information, for example, "Accounting."

**Description:** Type an optional long description of a data source name, for example, "My Accounting Database."

**Database Name:** Type the path to the directory that contains the database files. If none is specified, the current working directory is used. Click **Select** to browse the available FoxPro 3.0 database containers.

**Create Type:** You cannot change the Create Type for the FoxPro 3.0 DBC driver.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Locking:** Select the level of locking for the database file. FILE locks all of the records in the table. RECORD (the default) locks only the records affected by the statement. NONE offers the best performance but is intended only for single-user environments.

**Lock Compatibility:** Select the locking scheme the driver uses when locking records. Select Clipper, dBASE, Fox, Q+E, or Q+EVirtual. The default is Fox. 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. These values determine locking support as follows:

- 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.

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 are accessing 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 get corrupted.

**File Open Cache:** Type an integer value to specify the maximum number of used file handles to cache. For example, the value 4 specifies that when a user opens and closes four 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 increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

**Cache Size:** Type the number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 will not be able to see updates made by other users until you run the Select statement again. The default is 4.

**Data File Extension:** Type the file extension to use for data files. The default setting is DBF. The setting cannot be greater than three characters, and it cannot be one the driver already uses, such as MDX or CDX. The Data File Extension setting 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 attribute. The DataFileExtension value is used when no extension is specified.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

**Use Long Names:** Select this check box to use long file names as table names. The maximum table name length is specific to the environment in which you are running (for example, in Windows 9x, the maximum table name length is 128).

**Use Long Qualifiers:** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for path names is 128 characters.



**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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 improper environment or incorrect connection value, it will display 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.

---

## Defining Index Attributes

NOTE: This section does not apply to UNIX platforms. See [“Defining Index Attributes on UNIX” on page 108](#) for information on how to set index attributes on the UNIX platforms.

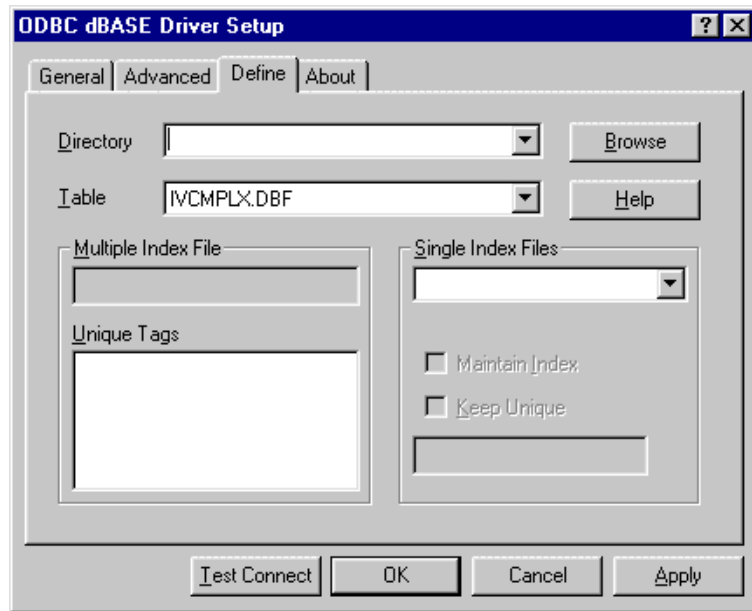


The Define tab of the ODBC dBASE Driver Setup dialog box allows you to define the attributes of index files. With the exception of Clipper, the family of databases that includes dBASE and FoxPro uses a multiple index file associated with a particular table (database file). This index file has a .MDX or .CDX extension and is automatically maintained by the driver. Tags within this index can be marked as unique.

Clipper used 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**.

**Directory:** Type the directory name that contains the table (database file). To display a directory listing, click **Browse**.

**Table:** Type the name of the table that contains the database information. To display a directory listing, click **Browse**.

**Multiple Index File:** Type the name of any multiple index file (with a .CDX extension or .MDX extension) associated with the table will be displayed in this field. This index file cannot be marked as unique, but tags within it can be.

**Unique Tags:** Tags associated with the multiple index file will appear in the list. To mark a tag as unique, single-click it; it will remain selected until you single-click it again. You can mark multiple tags in this manner.

NOTE: The Single Index Files pane is active only if you have selected a Clipper table.

**Single Index Files:** Select the file from the drop-down file 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



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 MERANT dBASE driver. The driver must be notified that index tags are unique. No configuration is needed for unique indexes that were created using the MERANT dBASE driver. When using files that were not created with the MERANT 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 plain 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 should be included.
- 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 table, the QEDBF.INI would be defined as:

```
[accts.dbf]
NUMUNIQUE=2
UNIQUE0=accts.mdx,ACCT_NAME
UNIQUE1=accts.mdx,ACCT_ID
```

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for dBASE is:

```
DSN=DBASE FILES;LCK=NONE;IS=0
```

[Table 4-1](#) gives the long and short names for each attribute, as well as a description.




To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, “The UNIX Environments” on page 473](#) for information about this file.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 4-1. dBASE Connection String Attributes**


---

Attribute	Description
 AppCodePage (ACP)	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>

**Table 4-1. dBASE Connection String Attributes** (cont.)

Attribute	Description
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CacheSize (CSZ)	<p>The number of 64KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 will not be able to see updates made by other users until you run the Select statement again.</p> <p>The initial default is 4.</p>
CreateType (CT)	<p>CreateType={dBASE4   dBASE5   Clipper   FoxPro25   FoxPro30}. Specifies the type of table or index to be created on a Create Table or Create Index statement.</p> <p>The initial default is dBASE5.</p>
Database (DB)	<p>The directory in which the dBASE files are stored.</p>
DataFileExtension (DFE)	<p>A string of three or fewer characters that specifies the file extension to use for data files. This value is used for all Create Table statements. If you execute a Create Table statement that uses an extension other than the one specified as the DataFileExtension, an error occurs.</p> <p>In other SQL statements, such as Select or Insert, you can specify an extension other than the DataFileExtension value. If you do not specify an extension value in these cases, the DataFileExtension value is used.</p> <p>The initial default is DBF.</p>
DataSourceName (DSN)	<p>A string that identifies a dBASE data source configuration in the system information. Examples include "Accounting" or "dBASE Files."</p>

**Table 4-1. dBASE Connection String Attributes** (cont.)

Attribute	Description
ExtensionCase (EC) 	<p>ExtensionCase={LOWER   UPPER}. Specifies whether upper- or lowercase file extensions are accepted.</p> <p>When set to LOWER, lowercase extensions are accepted.</p> <p>When set to UPPER (the initial default), uppercase extensions are accepted.</p>
FileOpenCache (FOC)	<p>An integer value that determines the maximum number of used file handles to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open.</p> <p>The initial default is 0, which means no file open caching.</p>
IntlSort (IS)	<p>IntlSort={0   1}. Determines the order in which records are retrieved when you issue a Select statement with an Order By clause.</p> <p>When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p>When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>



**Table 4-1. dBASE Connection String Attributes** (cont.)

Attribute	Description
LockCompatibility (LCOMP)	<p>LockCompatibility={Clipper   dBASE   Fox   Q+E   Q+EVirtual}. Specifies the locking scheme to be used in your dBASE tables. 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.</p> <p>When set to Clipper, Clipper-compatible locking is specified.</p> <p>When set to dBASE (the initial default), Borland-compatible locking is specified.</p> <p>When set to Fox, FoxPro-compatible locking is specified.</p> <p>When set to Q+E, locks are 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.</p> <p>When set to Q+EVirtual, locks are 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.</p> <p>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 are accessing 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 don't have to set LCOMP=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 LCOMP=Fox to ensure that your data does not get corrupted.</p>
Locking (LCK)	<p>Locking={NONE   RECORD   FILE}. Determines the level of locking for the database tables.</p> <p>When set to NONE, the best performance is offered, but is intended only for single-user environments.</p> <p>When set to RECORD (the initial default), only the records affected by the statement are locked. This is the initial default for all platforms.</p> <p>When set to FILE, all of the records in the table are locked.</p>

**Table 4-1. dBASE Connection String Attributes** (cont.)

Attribute	Description
UltraSafeCommit (USF)	<p>UltraSafeCommit={0   1}. Specifies when the driver flushes the file cache.</p> <p>When set to 0 (the initial default), the driver updates the directory entry when the file is closed. In this case, a machine "crash" before closing the file causes newly inserted records to be lost.</p> <p>When set to 1, the driver updates directory entries after each Commit. This decreases performance.</p>
UseLongNames (ULN)	<p>UseLongNames={0   1}. Determines whether the driver uses long file names as table names.</p> <p>When set to 0 (the initial default), the driver does not use long file names.</p> <p>When set to 1, the driver uses long file names. The maximum table name length is specific to the environment in which you are running (for example, in Windows 9x, the maximum table name length is 128).</p>
UseLongQualifiers (ULQ)	<p>UseLongQualifiers={0   1}. Determines whether the driver uses long path names as table qualifiers.</p> <p>When set to 0 (the initial default), the driver does not use long path names (the maximum path name length is 128 characters).</p> <p>When set to 1, the driver uses long path names (the maximum path name length is 255 characters).</p>

---

## Data Types

[Table 4-2](#) shows how dBASE data types map to the standard ODBC data types. These dBASE data types can be used in a Create Table statement. See [Appendix A, "SQL for Flat-File Drivers"](#) on [page 393](#) for the syntax of the Create Table statement.

[Table 4-3](#) shows how the additional FoxPro 3.0 tables and database containers 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 4-2. dBASE Data Types**

---

<b>dBASE</b>	<b>ODBC</b>
Binary <sup>1</sup>	SQL_LONGVARBINARY
Char <sup>2</sup>	SQL_CHAR
Date	SQL_TYPE_DATE
Float <sup>3</sup>	SQL_DECIMAL
General <sup>4</sup>	SQL_LONGVARBINARY
Logical	SQL_BIT
Memo	SQL_LONGVARCHAR
Numeric	SQL_DECIMAL

---

<sup>1</sup> dBASE V only

<sup>2</sup> 254 characters maximum (1024 for Clipper)

<sup>3</sup> dBASE IV and V only

<sup>4</sup> FoxPro and dBASE V only

---

---

**Table 4-3. Additional FoxPro 3.0 Data Types**

---

<b>FoxPro 3.0</b>	<b>ODBC</b>
Character (binary)	SQL_CHAR
Currency	SQL_DOUBLE
Datetime	SQL_TYPE_TIMESTAMP
Double	SQL_DOUBLE
Integer	SQL_INTEGER
Memo (binary)	SQL_LONGVARBINARY

---

## 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 [Appendix A, “SQL for Flat-File Drivers” on page 393](#) 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 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. The Create Index statement is used to create indexes and the Drop Index statement is used to delete indexes.

## Create Index

The type of index you create is determined by the value of the CreateType attribute, which you set in the setup dialog box (for UNIX, edit the system information file) or as a connection string option. 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]]...)
```

*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.

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.

*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 4-4](#) 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 4-4. 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*	Yes	Yes	240	255	No	Yes
FoxPro .CDX	Yes	Yes	240	255	Yes	Yes

\* 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.NDX, 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 (.DBT or .FPT) file

---

# SQL Statements for FoxPro 3.0 Database Containers

The FoxPro DBC driver supports four additional SQL statements:

- Create Database
- Add Table
- Remove Table
- Use

To create a new FoxPro 3.0 database container, use

```
CREATE DATABASE database_name
```

To add an existing table to the database container, use

```
ADD TABLE table_name
```

To remove a table from the database container (not delete the table, but unlink it from the database container), use

```
REMOVE TABLE table_name
```

To set the current database container to an existing database container, use

```
USE database_name
```

To add or delete columns from a table in a database container, use the Alter Table statement (see [“Alter Table Statement” on page 118](#)).

## 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 tables 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 table.

With file locking, all the records in the table 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.

## Using Locks on Local Files

If you use database locking and are accessing files locally (not on a network), run the DOS utility SHARE.EXE before running Windows 9x. If you add SHARE.EXE to your AUTOEXEC.BAT file, it runs automatically each time you boot your computer.

## 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 (see your server documentation).

If you are accessing a dBASE file locally, the limit depends on the buffer space allocated when SHARE.EXE was loaded (see 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. See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the dBASE driver. In addition, the following function is supported: SQLSetPos.

The dBASE driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

dBASE supports multiple connections and multiple statements per connection.

# 5 Connect ODBC for Excel Workbook

Connect ODBC for Excel Workbook (the "Excel Workbook driver") is supported in the Windows environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the Excel Workbook driver.

---

## Driver Requirements

The Excel Workbook driver requires the OLE32.DLL. This DLL is typically provided by the Windows System. It is also provided with Excel version 5 or 7. The Excel Workbook driver does not require Excel 5 or 7 to read .XLS files. To modify the files, however, you must use the Excel application.

---

## Using an Excel Database

An Excel workbook database is any Excel 5 or 7 or .XLS workbook file that contains one or more named lists containing record data. Each named list is treated as a table within the workbook database. The lists must be set up as follows:

- The first row in each list must contain the column labels that identify the fields in each record.

- The name of each list must be a book-level name, not a sheet-level name.
- Although the Excel Workbook driver recognizes one list named "Database" per .XLS file, it is recommended that you avoid using this name—both to ensure that the Excel Workbook driver recognizes other lists in the file and to prevent future naming conflicts.

To name a table from within Excel, highlight all the columns that are part of the table. After the columns are highlighted, from the menu bar select **Insert / Name / Define**. Type the table name in the dialog box to describe the highlighted table.

---

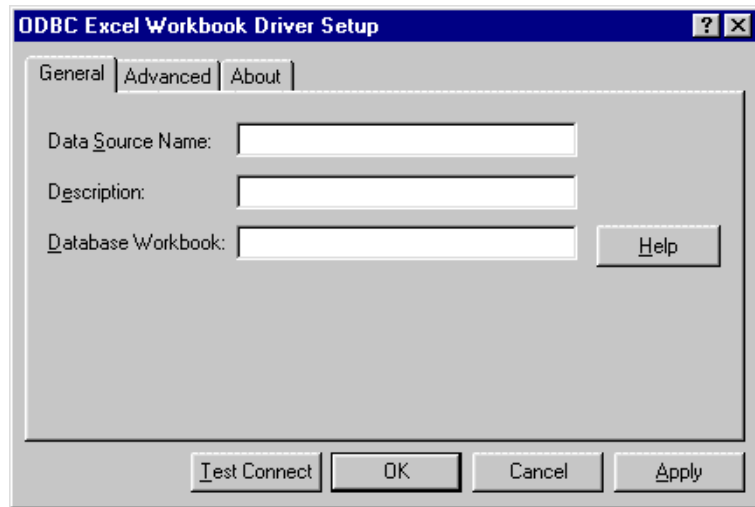
## Configuring Data Sources

Data sources are configured and modified through the ODBC Administrator. To configure an Excel data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Excel Workbook Driver Setup dialog box.

If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select the Excel Workbook driver and click **Finish** to display the ODBC Excel Workbook Driver Setup dialog box.





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.

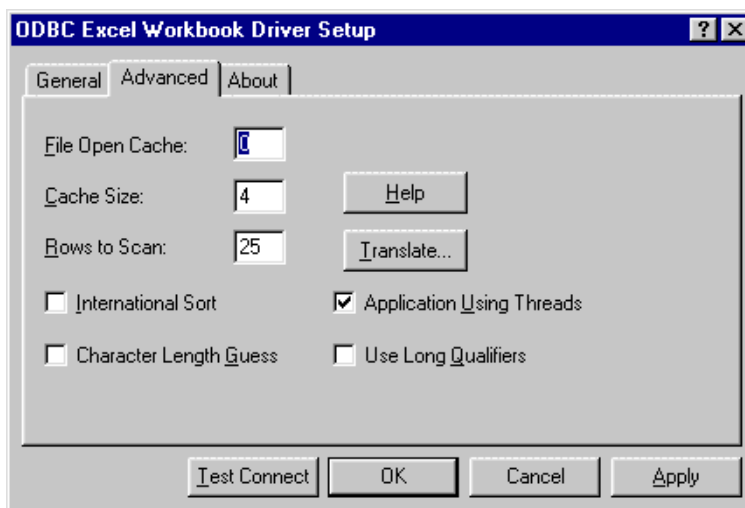
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this Excel data source configuration in the system information. Examples include "Accounting" or "Excel Database."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Excel .XLS file data."

**Database Workbook:** Type a name that identifies the workbook file containing the Excel database.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**File Open Cache:** Type an integer value that specifies the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four 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 increased performance. The default is 0, which means no file open caching.

**Cache Size:** Type the number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. The maximum number of blocks you can specify depends on the system memory available. The value must be a multiple of 64. The default is 4 (4 blocks of 64K, which equals 256K). If the cache size is greater than 0, when browsing backwards, you will not be able to see

updates made by other users until you run the Select statement again.

**Rows to Scan:** Type an integer value that specifies the number of rows to scan to determine the column data types. The default is 25 rows. A value of 0 means to scan the whole table.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

**Character Length Guess:** Select this check box to determine whether the driver tries to guess the length of character columns. If the check box is cleared (the default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255. If set to 1, the driver tries to guess the length.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Use Long Qualifiers:** Select this check box to use long pathnames as table qualifiers. When you select this check box, pathnames can be up to 255 characters. The default length for pathnames is 128 characters.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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 improper environment or incorrect connection value, it will display 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 default values 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.

---

# Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Excel is:

```
DSN=EXCEL FILES;FOC=4
```

[Table 5-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 5-1. Excel Connection String Attributes**

Attribute	Description
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CacheSize (CSZ)	<p>The number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 will not be able to see updates made by other users until you run the Select statement again.</p> <p>The initial default is 4.</p>
CharacterLengthGuess (CLG)	<p>CharacterLengthGuess={0   1}. Determines whether the driver tries to guess the length of character columns.</p> <p>When set to 0 (the initial default), the driver does not try to guess the length of character columns; instead, it assumes that all character columns have a length of 255.</p> <p>When set to 1, the driver tries to guess the length.</p>
Database (DB)	<p>The name of the .XLS workbook file containing the Excel database.</p>
DataSourceName (DSN)	<p>A string that identifies an Excel data source configuration in the system information. Examples include "Accounting" or "Excel Files."</p>
FieldGuessing (FG)	<p>FieldGuessing={0   1}. Determines whether the driver tries to guess column data types.</p> <p>When set to 0, the driver assumes that all data types are SQL_CHAR.</p> <p>When set to 1 (the initial default), the driver attempts to guess the data types.</p>

**Table 5-1. Excel Connection String Attributes** (cont.)

<b>Attribute</b>	<b>Description</b>
FileOpenCache (FOC)	<p>The maximum number of unused file opens to cache. For example, when FileOpenCache=4, and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open.</p> <p>The initial default is 0, which means no file open caching.</p>
IntlSort (IS)	<p>IntlSort={0   1}. Determines the order in which records are retrieved when you issue a Select statement with an Order By clause.</p> <p>When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p>When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>
ScanRows (SR)	<p>The number of rows to scan to determine the column data types. A value of 0 means to scan the whole table.</p> <p>The initial default is 25.</p>
UseLongQualifiers (ULQ)	<p>UseLongQualifiers={0   1}. Determines whether the driver uses long pathnames as table qualifiers.</p> <p>When set to 0 (the initial default), the driver does not use long pathnames (the maximum path name length is 128 characters).</p> <p>When set to 1, the driver uses long pathnames (the maximum path name length is 255 characters).</p>

---

## Data Types

[Table 5-2](#) shows how Excel data types map to the standard ODBC data types.

---

**Table 5-2. Excel Data Types**

---

Excel	ODBC
Char	SQL_VARCHAR
Date	SQL_TYPE_DATE
Logical	SQL_BIT
Number Float	SQL_DOUBLE
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP

---

## SQL Supported

The Excel Workbook driver supports only SQL Select statements. All of the Select statement clauses described in [Appendix A, “SQL for Flat-File Drivers” on page 393](#) are supported by Excel Select statements.

The Excel Workbook driver does not support the following SQL statements: Insert, Update, Delete, Create Table, and Drop Table.

### Table and Column Names

Table names are case-sensitive when using the Excel Workbook driver.

Excel files contain column names in the first row of the database section or named list that is used as a table. Use these names as



the column names in a Select statement. Column names are limited to 32 characters. If column names are lowercase, contain upper- and lowercase, contain blank spaces, or are reserved words, surround them with the grave character ( ` ) (ASCII 96).

For example:

```
SELECT `name` FROM emp
```

The Excel Workbook driver does not support the Create Table statement.

---

## ODBC Conformance Level

See [Appendix C, "ODBC API and Scalar Functions" on page 427](#) for a list of the API functions supported by the Excel Workbook driver. In addition, the following function is supported: SQLSetPos.

The Excel Workbook driver supports backward and random fetching in SQLExtendedFetch or SQLFetchScroll. The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The Excel Workbook driver supports multiple connections and multiple statements per connection.



# 6 Connect ODBC for Informix

Connect ODBC for Informix (the "Informix driver") supports multiple connections to the Informix database system versions 7.x and 9.x in the Windows and UNIX (not including Linux) environments. See "[Environment-Specific Information](#)" on [page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the Informix driver.

---

## Driver Requirements

This section provides the system requirements for using the Informix driver on all supported platforms.

### Windows

To access remote Informix 7.x or 9.x databases through the Informix driver, you need one of the following:

- INFORMIX-Connect for Windows platforms, version 2.x
- INFORMIX-Client Software Developer's Kit for Windows platforms, version 2.x

**NOTE:** The DataDirect Informix driver does not work with versions earlier than 9.1.3 of the previous INFORMIX-Connect product.

Use the **Setnet32** utility supplied by Informix to define servers and the location of the INFORMIX directory. Use **llogin** 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, 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.

You need INFORMIX-Client Software Developer's Kit for UNIX platforms, version 2.x, from Informix to access remote Informix 7.x or 9.x databases through the Informix driver.

**NOTE:** The DataDirect Informix driver for Solaris and HP-UX does not work with versions earlier than 9.1.3 of the previous products, INFORMIX-Connect and ESQ/C. The driver for AIX does not work with versions earlier than 9.1.4 of INFORMIX-Connect and ESQ/C.

---

# Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.

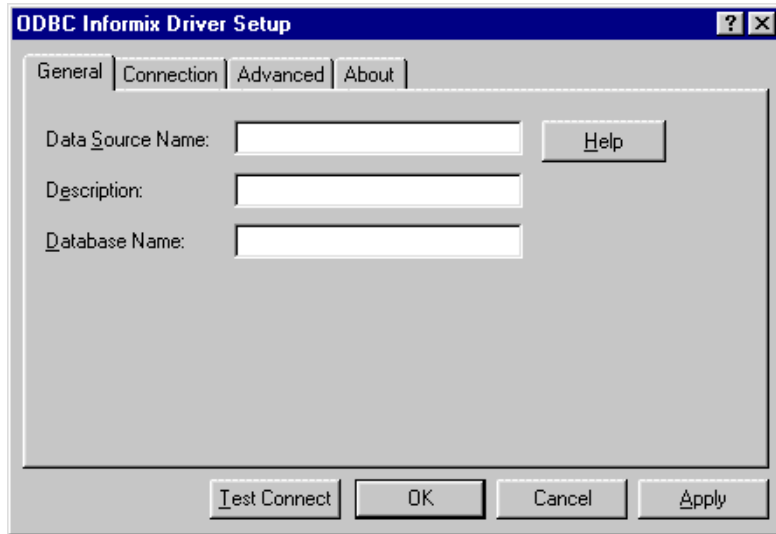


In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 6-1 on page 149](#). You must also edit this file to perform a translation. See [Appendix H, “The UNIX Environments” on page 473](#) for information about editing the file.

## To configure an Informix data source on Windows:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Informix Driver Setup dialog box.

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



**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.

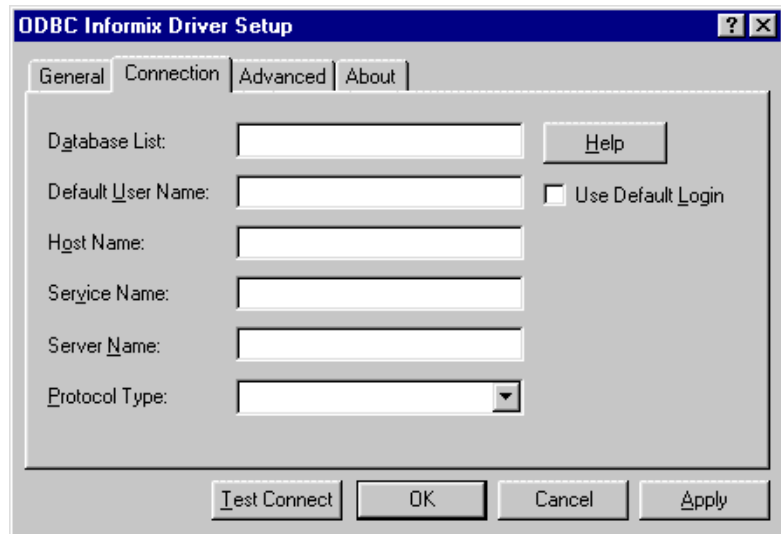
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this Informix data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Informix 7 files on Server number 1."

**Database Name:** Type the name of the database to which you want to connect by default.

- 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 any of the following optional information; then, click **Apply**.

**Database List:** Type a list of databases that will be displayed in the Logon dialog box if **Get DB List From Informix** on the Advanced tab is *not* selected. Separate multiple values with commas (for example, db1, db2, db3).

**Default User Name:** Type the name of the user as specified on the Informix server.

**Use Default Login:** Select this check box to read the Logon ID and Password entries directly from the Informix registry. The check box is not selected by default; that is, logon information is read from the system information, the connection string, or the Logon to Informix dialog box.

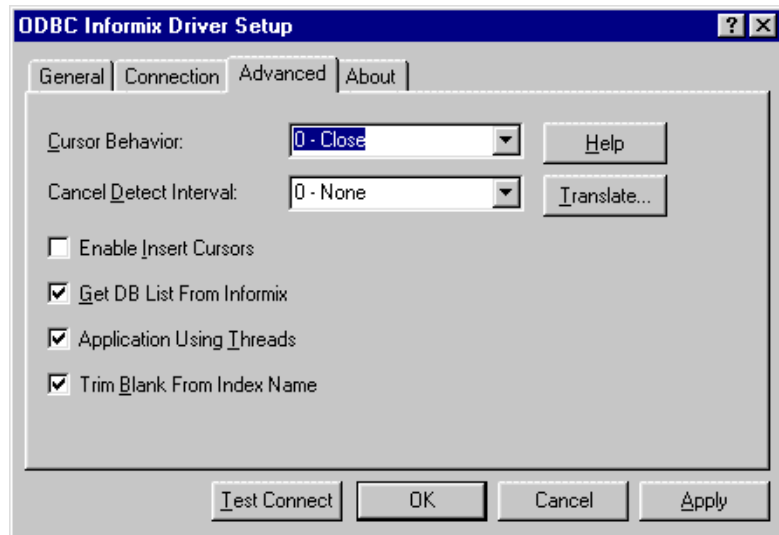
**Host Name:** Type the name of the machine on which the Informix server resides.

**Service Name:** Type the name of the service as it appears on the host machine. This service is assigned by the system administrator. The name you specify is displayed in the Informix Server Options dialog box.

**Server Name:** Type the name of the Informix server as it appears in the sqlhosts file.

**Protocol Type:** Select the protocol used to communicate with the server.

- 5 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

**Cursor Behavior:** Select Preserve to hold the cursor at the current position when the transaction ends. Otherwise, leave this field set to Close. Selecting Preserve may impact the performance of your database operations.

**Cancel Detect Interval:** Select a value other than "0 - None" to allow long-running queries in threaded applications to be canceled if the application issues a SQLCancel. The value you



`select` determines how often (in seconds) the driver checks whether a query has been canceled using `SQLCancel`. If the driver determines that `SQLCancel` has been issued, the query is canceled. For example, if you select 5, then for every pending query, the driver checks every five seconds to see whether the application has canceled execution of the query using `SQLCancel`. The default is 0, which means that queries are not canceled even if a `SQLCancel` is issued.

**Enable Insert Cursors:** Select this check box to determine whether the driver can use Insert cursors during inserts governed by parameters. Using Insert cursors improves performance during multiple Insert operations using the same statement. This option enables insert data to be buffered in memory before being written to disk. When this check box is cleared (the default), the driver does not use Insert cursors.

**Get DB List From Informix:** Select this check box to determine whether the driver requests the database list to be returned from the Informix server or from the database list that the user entered during driver setup.

When the check box is selected (the default), the driver requests the database list from the Informix server. When cleared, the driver uses the list that was entered by the user at driver setup.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Clearing this check box avoids the additional processing required for ODBC thread-safety standards.

**Trim Blank From Index Name:** Select this check box to specify whether the leading space should be trimmed from a system-generated index name. This option is provided to address problems with applications that cannot process a

leading space in index names. When this box is selected (the default), the driver trims the leading space.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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 is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box" on page 147](#) 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 will display 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].
```

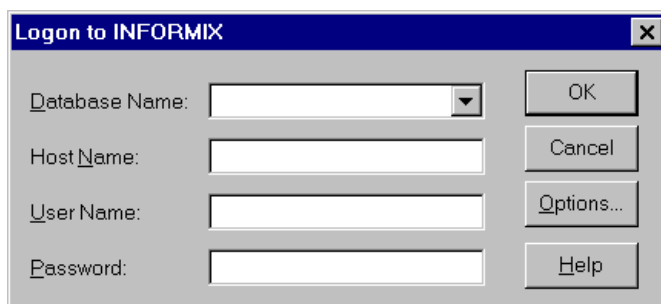
Click **OK**.

- 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.


---

## Connecting to a Data Source Using a Logon Dialog Box

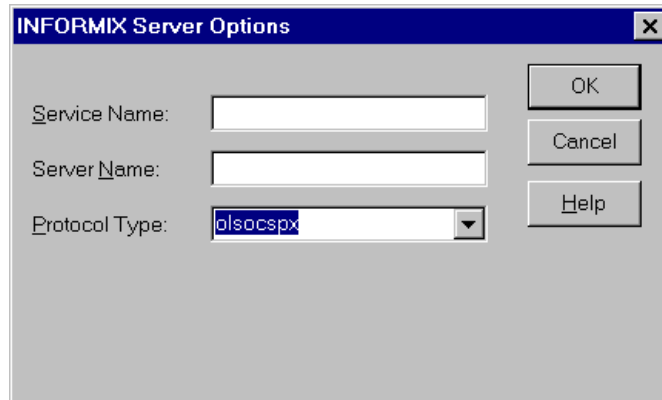
Some ODBC applications display a logon dialog box when you are connecting to a data source. In these cases, the data source name has already been specified. The dialog box is as follows:



In this dialog box, perform the following steps:

- 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.
- 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  Optionally, 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 system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Informix is:

```
DSN=INFORMIX TABLES;DB=PAYROLL
```

[Table 6-1](#) gives the long and short names for each attribute, as well as a description.




To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, “The UNIX Environments” on page 473](#) for information about this file.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 6-1. Informix Connection String Attributes**




---

Attribute	Description
AppCodePage (ACP) 	Valid values for this attribute are listed in <a href="#">Appendix I</a> . The code page that you specify must be the same as the code page used by your application.  The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order: <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).


**Table 6-1. Informix Connection String Attributes** (cont.)

Attribute	Description
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CancelDetectInterval (CDI)	<p>A value in seconds that determines how often the driver checks whether a query has been canceled using SQLCancel. If the driver determines that SQLCancel has been issued, the query is canceled. This attribute determines whether long-running queries in threaded applications are canceled if the application issues a SQLCancel. If set to 0 (the initial default), queries are not canceled even if SQLCancel is issued.</p> <p>For example, if CancelDetectInterval is set to 5, then for every pending request, the driver checks every five seconds to see whether the application has canceled execution of the query using SQLCancel.</p>
CursorBehavior (CB)	<p>CursorBehavior={0   1}. Determines whether cursors will be preserved or closed at the end of each transaction.</p> <p>When set to 0 (the initial default), cursors will be closed at the end of each transaction.</p> <p>When set to 1, cursors will be held at the current position when the transaction ends. This value may impact the performance of your database operations.</p>
Database (DB)	<p>The name of the database to which you want to connect.</p>
DataSourceName (DSN)	<p>A string that identifies an Informix data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."</p>

**Table 6-1. Informix Connection String Attributes** (cont.)

Attribute	Description
EnableInsertCursors (EIC)	<p>EnableInsertCursors={0   1}. Determines whether the driver can use Insert cursors during inserts governed by parameters.</p> <p>When set to 0, the driver does not use Insert cursors.</p> <p>When set to 1 (the initial default), the driver uses Insert cursors. Using Insert cursors improves performance during multiple Insert operations using the same statement. This option enables insert data to be buffered in memory before being written to disk.</p>
GetDBListFromInformix (GDBLFI)	<p>GetDBListFromInformix={0   1}. 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.</p> <p>When set to 0, the driver uses the list that was entered by the user at driver setup.</p> <p>When set to 1 (the initial default), the driver requests the database list from the Informix server.</p>
HostName (HOST)	The name of the machine on which the Informix server resides.
	
LogonID (UID)	Your user name as specified on the Informix server.
Password (PWD)	A password.
Protocol (PRO)	<p>Protocol={olsocsp   olsoctcp   onsocsp   onsoctcp   seipcpip   sesocsp   sesoctcp}. Specifies the protocol used to communicate with the server. You can specify one or more values; separate the names with commas.</p>
	
ServerName (SRVR)	The name of the server running the Informix database.
Service (SERV)	The name of the service as it appears on the host machine. This service is assigned by the system administrator.
	

**Table 6-1. Informix Connection String Attributes** (cont.)

Attribute	Description
TrimBlankFrom IndexName (TBFIN)	<p>TrimBlankFromIndexName={0   1}. Determines whether the leading space should be trimmed from a system-generated index name. This option is provided to address problems with applications that cannot process a leading space in index names.</p> <p>When set to 0, the driver does not trim the space.</p> <p>When set to 1 (the initial default), the driver trims the leading space.</p>
UseDefaultLogin (UDL) 	<p>UseDefaultLogin={0   1}. Determines from where the logon information is read.</p> <p>When set to 0 (the initial default), logon information is read from the system information, the connection string, or the Logon to Informix dialog box.</p> <p>When set to 1, the Logon ID and Password are read directly from the Informix registry.</p>

---

## Data Types

[Table 6-2](#) shows how the Informix data types map to the standard ODBC data types.



**Table 6-2. Informix Data Types**

<b>Informix</b>	<b>ODBC</b>
Blob	SQL_LONGVARBINARY
Boolean	SQL_BIT
Byte <sup>1</sup>	SQL_LONGVARBINARY
Char	SQL_CHAR
Clob	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Datetime year to fraction(5)	SQL_TYPE_TIMESTAMP
Datetime year to fraction(f) <sup>2</sup>	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) <sup>2</sup>	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) <sup>2</sup>	SQL_INTERVAL_DAY_TO_SECOND
Interval hour(p) to hour	SQL_INTERVAL_HOUR
Interval hour(p) to minute	SQL_INTERVAL_HOUR_TO_MINUTE

<sup>1</sup>Not supported for Standard Engine Databases

<sup>2</sup>Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 5.

**Table 6-2. Informix Data Types** (cont.)

<b>Informix</b>	<b>ODBC</b>
Interval hour(p) to second	SQL_INTERVAL_HOUR_TO_SECOND
Interval hour(p) to fraction(f) <sup>2</sup>	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) <sup>2</sup>	SQL_INTERVAL_MINUTE_TO_SECOND
Interval second(p) to second	SQL_INTERVAL_SECOND
Interval second(p) to fraction(f) <sup>2</sup>	SQL_INTERVAL_SECOND
Interval fraction to fraction(f) <sup>2</sup>	SQL_VARCHAR
Lvarchar	SQL_VARCHAR
Money	SQL_DECIMAL
Serial	SQL_INTEGER
Serial8	SQL_BIGINT
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text <sup>1</sup>	SQL_LONGVARCHAR
Varchar <sup>1</sup>	SQL_VARCHAR

<sup>1</sup>Not supported for Standard Engine Databases

<sup>2</sup>Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 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).

---

# MTS Support

The Informix driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC) using the XA Protocol. To enable the DTC support, the clients must be INFORMIX-Connect version 2.20 or higher. For a general discussion of MTS and DTC, refer to the help file of the "Microsoft Transaction Server SDK."

## 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

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

- 1 Turn on **STATIC** cursors. For example:

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

- 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 regarding Step 1:** 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.

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, **SQL\_PERSIST\_AS\_XML**. A client application must call **SQLSetStmtAttr** with this new attribute as an argument. See the following table for the definition of valid arguments for **SQLSetStmtAttr**.

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

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

Function Sequence Error

---

## 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.

See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the Informix driver. In addition, 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.

---

## Number of Connections and Statements Supported

The Informix driver supports multiple connections and multiple statements per connection to the Informix database system.

# 7 Connect ODBC for Informix Dynamic Server

Connect ODBC for Informix Dynamic Server (the "Informix Dynamic Server driver") supports multiple connections to the Informix database system version 9.x in the Windows and UNIX environments. See "[Environment-Specific Information](#)" on [page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the Informix driver.

---

## Driver Requirements

There are no database client requirements for the Informix Dynamic Server driver.

---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.



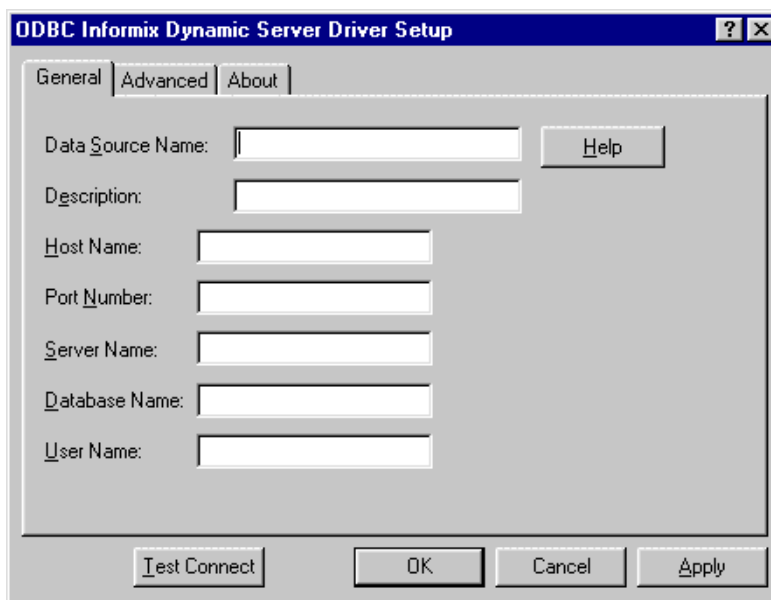
In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 7-1 on page 166](#). You must also edit this file to perform a translation.

See [Appendix H, “The UNIX Environments”](#) on page 473 for information about editing the file.

**To configure an Informix data source on Windows:**

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Informix Dynamic Server Driver Setup dialog box.

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



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

**Data Source Name:** Type a string that identifies this Informix data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Informix 7 files on Server number 1."

**Host Name:** Type the name of the machine on which the Informix server resides.

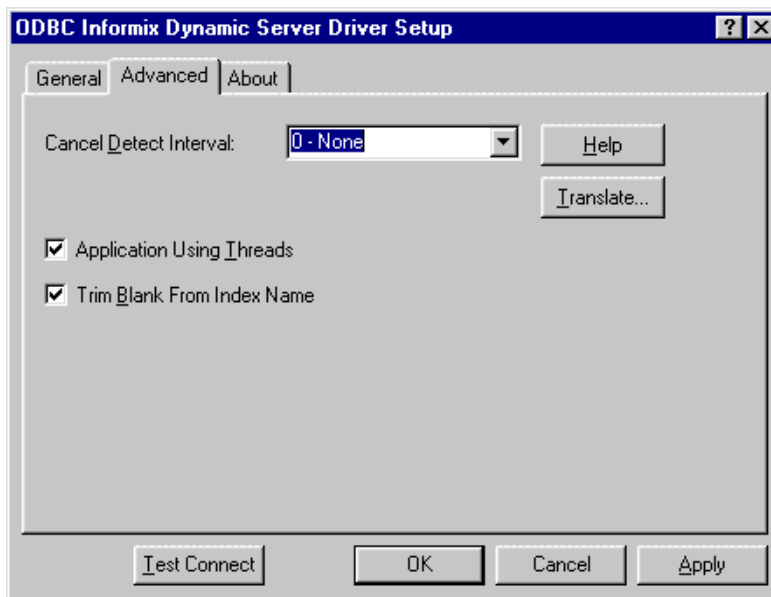
**Port Number:** Type the port number of the server listener.

**Server Name:** Type the name of the Informix server as it appears in the sqlhosts file.

**Database Name:** Type the name of the database to which you want to connect by default.

**User Name:** Type your user name as specified on the Informix server.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Cancel Detect Interval:** Select a value other than "0 - None" to allow long-running queries in threaded applications to be canceled if the application issues a SQLCancel. The value you select determines how often (in seconds) the driver checks whether a query has been canceled using SQLCancel. If the driver determines that SQLCancel has been issued, the query is canceled. For example, if you select 5, then for every pending query, the driver checks every five seconds to see whether the application has canceled execution of the query using SQLCancel. The default is 0, which means that queries are not canceled even if SQLCancel is issued.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Clearing this check box avoids

the additional processing required for ODBC thread-safety standards.

**Trim Blank From Index Name:** Select this check box to specify whether the leading space should be trimmed from a system-generated index name. This option is provided to address problems with applications that cannot process a leading space in index names. When this box is selected (the default), the driver trims the leading space.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. A logon dialog box is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box"](#) on page 164 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 will display 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].
```

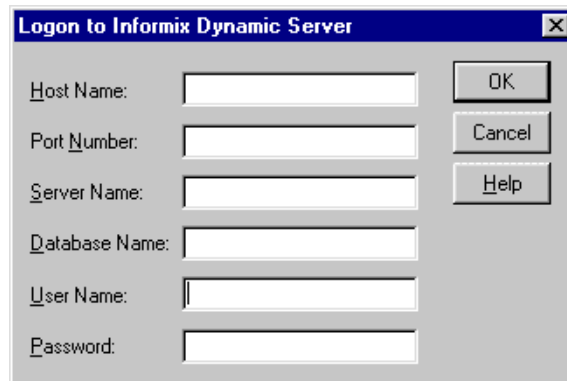
Click **OK**.

- 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.

---

## Connecting to a Data Source 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. The dialog box is as follows:



In this dialog box, perform the following steps:

- 1 In the Host Name field, type the name of the host machine on which the Informix server resides.
- 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 as it appears in the sqlhosts file.

- 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 system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Informix is:

```
DSN=INFORMIX TABLES;DB=PAYROLL
```

[Table 7-1](#) gives the long and short names for each attribute, as well as a description.




To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, “The UNIX Environments” on page 473](#) for information about this file.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 7-1. Informix Dynamic Server Connection String Attributes**


---

Attribute	Description
AppCodePage (ACP) 	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>
ApplicationUsing Threads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CancelDetect Interval (CDI)	<p>A value in seconds that determines how often the driver checks whether a query has been canceled using SQLCancel. If the driver determines that SQLCancel has been issued, the query is canceled. This attribute determines whether long-running queries in threaded applications are canceled if the application issues a SQLCancel. If set to 0 (the initial default), queries are not canceled even if SQLCancel is issued.</p> <p>For example, if CancelDetectInterval is set to 5, then for every pending request, the driver checks every five seconds to see whether the application has canceled execution of the query using SQLCancel.</p>

---

**Table 7-1. Informix Dynamic Server Connection String Attributes** (cont.)

---

<b>Attribute</b>	<b>Description</b>
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies an Informix data source configuration in the system information. Examples include "Accounting" or "INFORMIX-Serv1."
HostName (HOST) 	The name of the machine on which the Informix server resides.
LogonID (UID)	Your user name as specified on the Informix server.
Password (PWD)	A password.
PortNumber (PORT)	The port number of the server listener. There is no default value.
ServerName (SRVR)	The name of the server running the Informix database.
TrimBlankFrom IndexName (TBFIN)	TrimBlankFromIndexName={0   1}. Determines whether the leading space should be trimmed from a system-generated index name. This option is provided to address problems with applications that cannot process a leading space in index names.  When set to 0, the driver does not trim the space.  When set to 1 (the initial default), the driver trims the leading space.

---

## MTS Support

The Informix Dynamic Server 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."

## Data Types

Table 7-2 shows how the Informix data types map to the standard ODBC data types.

**Table 7-2. Informix Data Types**

<b>Informix</b>	<b>ODBC</b>
Blob	SQL_LONGVARBINARY
Boolean	SQL_BIT
Byte	SQL_LONGVARBINARY
Char	SQL_CHAR
Clob	SQL_LONGVARCHAR
Date	SQL_TYPE_DATE
Datetime year to fraction(5)	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(5)	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

\* Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 5.



**Table 7-2. Informix Data Types (cont.)**

<b>Informix</b>	<b>ODBC</b>
Interval day(p) to fraction(f)*	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)*	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)*	SQL_INTERVAL_MINUTE_TO_SECOND
Interval second(p) to second	SQL_INTERVAL_SECOND
Interval second(p) to fraction(f)*	SQL_INTERVAL_SECOND
Interval fraction to fraction(f)*	SQL_VARCHAR
Lvarchar	SQL_VARCHAR
Money	SQL_DECIMAL
Nchar	SQL_CHAR
Serial	SQL_INTEGER
Serial8	SQL_BIGINT
Smallfloat	SQL_REAL
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR

\*Fraction(f) types are mapped to fraction(5) in the driver. The precision is type dependent and the scale as 5.

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on `STATIC` cursors. For example:

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

- 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 regarding Step 1:** 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.

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, `SQL_PERSIST_AS_XML`. A client application must call `SQLSetStmtAttr` with this new attribute as an argument. See the following table for the definition of valid arguments for `SQLSetStmtAttr`.

Argument	Definition
<i>StatementHandle</i>	The handle of the statement that contains the result set to persist as XML.

Argument	Definition
<i>Attribute</i>	SQL_PERSIST_AS_XML. This new statement attribute can be found in the file qesqlx.h, which is installed with the driver.
<i>ValuePtr</i>	Pointer to a URL that specifies the full path name of the XML data file to be generated. The directory specified in the path name must exist, and if the specified file name exists, the file will be overwritten.
<i>StringLength</i>	The length of the string pointed to by ValuePtr or SQL_NTS if ValuePtr points to a null terminated string.

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

Function Sequence Error

---

## Isolation and Lock Levels Supported

Informix supports isolation levels 0 (read uncommitted), 1 (read committed), and 3 (serializable). The default is 1.

Informix supports record-level locking.

See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the Informix Dynamic Server driver.

The Informix Dynamic Server driver also supports the following functions:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The Informix Dynamic Server driver supports multiple connections and multiple statements per connection to the Informix database system.

# 8 Connect ODBC for MySQL

Connect ODBC for MySQL (the "MySQL driver") supports MySQL on the Linux distributions on UNIX. See ["Environment-Specific Information" on page 33](#) for detailed information about the Linux distributions supported by this driver.

See the README file shipped with your DataDirect product for the file name of the MySQL driver.

---

## Driver Requirements

You must have the MySQL client for Linux version 3.22.25 or higher installed.

---

## Configuring and Connecting to Data Sources

To configure a data source, you must edit the system information file using the long names of the attributes listed in [Table 8-1](#). See [Appendix H, "The UNIX Environments" on page 473](#) for details on configuring the system information file.

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information file to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

The connection string has the form:

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

An example of a connection string for MySQL is:

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

[Table 8-1](#) gives the name for each attribute as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 8-1. MySQL Connection String Attributes**

---

Attribute	Description
AppCodePage (ACP)	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies a MySQL data source configuration in the system information. Examples include "Accounting" or "MySQLServ1."
Hostname (HN)	The hostname of the server on which the database resides. The default is Hostname = localhost.

**Table 8-1. MySQL Connection String Attributes** (cont.)

Attribute	Description
LogonID (UID)	The default logon ID used to connect to your MySQL database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.
Password (PWD)	The password used to connect to your MySQL database.

## Data Types

[Table 8-2](#) shows how the MySQL data types are mapped to the standard ODBC data types.

**Table 8-2. MySQL Data Types**

MySQL	ODBC
Bigint	SQL_BIGINT
Blob	SQL_LONGVARCHAR
Char	SQL_CHAR
Date	SQL_TYPE_DATE
Datetime	SQL_TYPE_TIMESTAMP
Decimal	SQL_DECIMAL
Double	SQL_DOUBLE
Enum	SQL_CHAR
Float	SQL_FLOAT
Int	SQL_INTEGER
Integer	SQL_INTEGER
Longblob	SQL_LONGVARBINARY
Mediumblob	SQL_LONGVARBINARY
Mediumtext	SQL_LONGVARCHAR

---

**Table 8-2. MySQL Data Types (cont.)**


---

MySQL	ODBC
Middleint	SQL_INTEGER
Set	SQL_CHAR
Smallint	SQL_SMALLINT
Text	SQL_LONGVARCHAR
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Tinyblob	SQL_LONGVARBINARY
Tinyint	SQL_TINYINT
Varchar	SQL_VARCHAR
Year	SQL_SMALLINT

---

## ODBC Conformance Level

The MySQL driver supports the API functions listed in [Appendix C, “ODBC API and Scalar Functions” on page 427](#). In addition, the X/Open function `SQLPrimaryKeys` is supported.

The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The MySQL database system supports a single connection and multiple statements per connection.



## 9 Connect ODBC for Oracle

Connect ODBC for Oracle supports two separate drivers. Connect ODBC for Oracle (the "Oracle driver") supports the Oracle7 database system. The Oracle driver is supported in the Windows and UNIX (not including Linux or HP-UX 11 aCC) environments using the Oracle7 client.

Connect ODBC for Oracle8 (the "Oracle8 driver") supports the Oracle 8.0.5 and higher database system. It also supports the Oracle7 database system version 7.3.4 or higher when using the Oracle Net8 client. The Oracle8 driver is supported in the Windows and UNIX environments.

See "[Environment-Specific Information](#)" on page 33 for detailed information about the Windows and UNIX environments supported by these two drivers.

See the README file shipped with your DataDirect product for the file names of the Oracle drivers.

---

### Driver Requirements

This section provides the system requirements for using the Oracle drivers on both Windows and UNIX.



#### Windows

Both Oracle7 and Oracle8 client information for Windows 9x, Windows NT, and Windows 2000 is listed below.

**IMPORTANT:** You must have *all* components of the Oracle client software installed; otherwise, the driver will not operate properly.

## **Oracle7**

The Oracle SQL\*Net product is required to access remote Oracle7 databases. The appropriate DLLs for the current version of SQL\*Net and OCIW32.DLL must be on your path. For example, SQL\*Net 2.3 requires ORA73.DLL, CORE35.DLL, NLSRTL32.DLL, and CORE350.DLL, as well as OCIW32.DLL.

## **Oracle8**

The Oracle Net8 Client 8.0.5, or higher, is required to access remote Oracle8 databases (servers) 8.0.5 or higher. On Intel systems, the appropriate DLLs for the Oracle Net8 Client must be on your path, for example, ORA804.DLL, PLS804.DLL, and OCI.DLL.

The Oracle Net8 Client 8.0.5 or higher, is required to access remote Oracle databases (servers) 7.3.4.x.



## **UNIX**

Both Oracle and Oracle8 client information for UNIX is listed below.

**IMPORTANT:** You must have *all* components of the Oracle client software installed; otherwise, the driver will not operate properly. See the DataDirect README for a component list.

## ***Oracle and Oracle8***

Before you can use the Oracle data source, you must have Oracle SQL\*Net or Oracle Net8 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. Note that on Linux you must use the Oracle Net8 version 8.1.6.1.

Set the environment variable ORACLE\_HOME to the directory where you installed Oracle SQL\*Net or Oracle Net8. 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
```

## ***Building the Required Oracle7 SQL\*Net Shared Library***

The Oracle driver requires a one-time site linking to build an Oracle7 SQL\*Net shared library on AIX and, for Oracle7.1 only, on Solaris and HP-UX. This site linking binds your unique Oracle7 SQL\*Net configuration into the file, which is used by the Oracle driver to access local and remote Oracle databases.

Before you build the Oracle7 SQL\*Net shared library, install Oracle and set the environment variable ORACLE\_HOME to the directory where you installed Oracle. Connect ODBC provides the script, genclntsh, to build the Oracle7 SQL\*Net shared library. This script is in the src/oracle directory.

The following commands build the Oracle7 SQL\*Net shared library:

```
cd ${ODBC_HOME}/src/oracle
genclntsh
```

### ***Building the Required Oracle Net8 Shared Library on Solaris and HP-UX 11***

A one-time site linking to build a replacement Oracle Net8 shared library is required in the following cases:

- For Oracle Net8 Client 8.0.5 on Solaris and HP-UX 11
- For Oracle Net8 Client 8.1.5 and 8.1.6 on HP-UX 11

This site linking binds your unique Oracle Net8 configuration into the file, which is used by the Oracle driver to access local and remote Oracle databases.

The Oracle8 driver requires the shared library `libclntsh.so` (Solaris) and `libclntsh.sl` (HP-UX 11), which is built by the Oracle script `genclntsh`. The `genclntsh` script provided by Oracle causes errors resulting from undefined symbols. Users running Oracle8 on specific platforms with certain versions of Oracle client software (as documented below) must therefore use the `genclntsh8` or `genclntsh816` script provided with Connect ODBC for Oracle to build a replacement `libclntsh.so` or `libclntsh.sl`. These scripts, in the `src/oracle` directory, place the new `libclntsh.so` or `libclntsh.sl` in `../lib`, which is your `$ODBC_HOME/lib` directory; it does not overwrite the original `libclntsh.so` or `libclntsh.sl` in the `$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 8.0.5 on Solaris and Oracle Net8 Client 8.0.5 and 8.1.5 on HP-UX 11, the following commands build the Oracle Net8 shared library:

```
cd ${ODBC_HOME}/src/oracle
genclntsh8
```

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:** Oracle8 users will have the original libclntsh.so library in the \$ORACLE\_HOME/lib directory. Therefore, the \$ODBC\_HOME/lib directory, containing the correct library, *must* be on the LD\_LIBRARY\_PATH (Solaris) or SHLIB\_PATH (HP-UX 11) *before* \$ORACLE\_HOME/lib. Otherwise, the original Oracle library will be loaded, resulting in the unresolved symbol error.

---



---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.



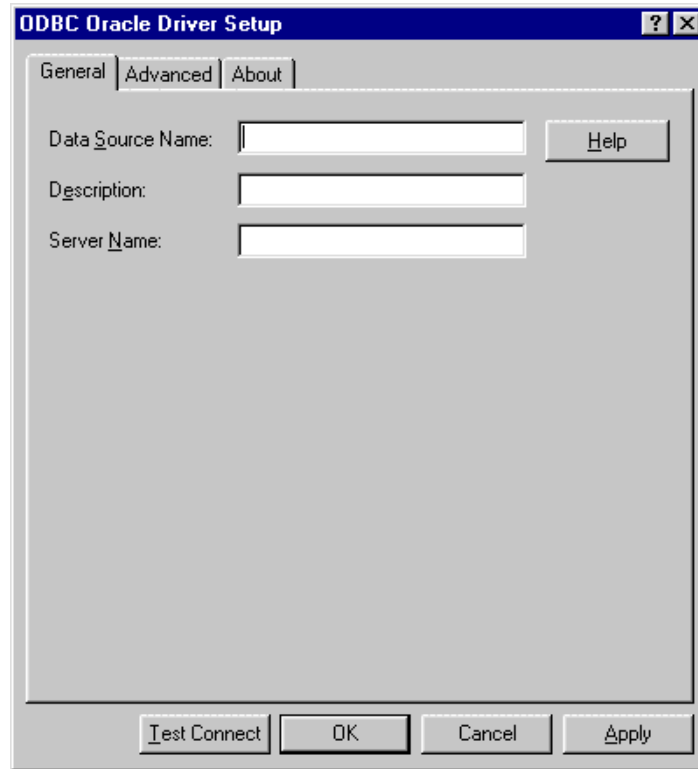
In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 9-1 on page 189](#). You must also edit this file to perform a translation. See [Appendix H, "The UNIX Environments," on page 473](#) for information about editing the file.

### To configure an Oracle data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Oracle Driver Setup dialog box.

If you are configuring a new data source, click **Add** to display a list of installed drivers. Select the Oracle driver of your

choice and click **Finish** to display the ODBC Oracle Driver Setup dialog box.



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.

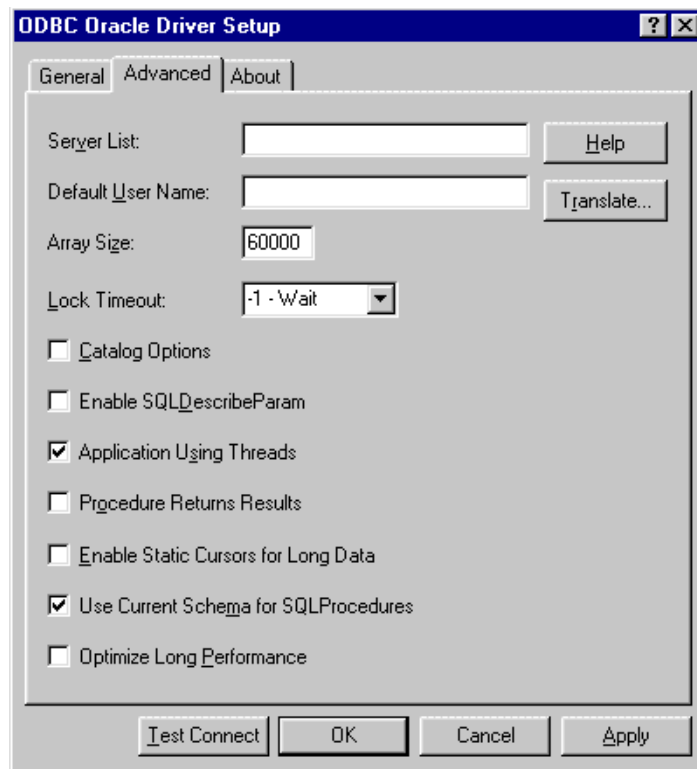
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this Oracle data source configuration in the system information. Examples include "Accounting" or "Oracle-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Oracle on Server number 1."

**Server Name:** Type the client connection string designating the server and database to be accessed. The information required varies depending on the client driver you are using. ["Connecting to a Data Source Using a Connection String" on page 188](#) describes the format of the connection string.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Server List:** Type a list of client connection strings that will appear in the logon dialog box. 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 User Name:** Type the default user name used to connect to your Oracle database. A default user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the logon dialog box or connection string.

**Array Size:** Type the number of bytes the driver uses for fetching multiple rows. Values can be an integer from 1 up to 4 GB; the default is 60000. 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.

**Lock Timeout:** Select a value of 0 or -1 that specifies whether Oracle should wait for a lock to be freed before raising an error when processing a Select...For Update statement. Values can be -1 (wait forever) or 0 (do not wait). The default is -1.

**Packet Size (Oracle7 Only):** Select a value that controls the packet size for TCP/IP connections. Specify one of the following packet sizes: 1024, 2048, 4096, or 8192. Any other value is ignored.

The Packet Size option is used only when the connection string specified in the Server Name option is T for TCP/IP as the driver prefix. See the ServerName option described in [Table 9-1 on page 189](#).

**Catalog Options:** Select this check box if you want the result column REMARKS for the catalog functions SQLTables and SQLColumns, and the result column COLUMN\_DEF for the catalog function SQLColumns to have meaning for Oracle. Selecting this box reduces the performance of your queries. By default, the check box is not selected, which returns SQL\_NULL\_DATA for the result columns COLUMN\_DEF and REMARKS.

**Enable SQLDescribeParam:** Select this check box to enable the SQLDescribeParam function, which results in all parameters being described with a data type of SQL\_VARCHAR. This



option should be selected when using Microsoft Remote Data Objects (RDO) to access data. By default, the check box is not selected.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. When using the driver with single-threaded applications, clear this check box. Turning off this setting avoids additional processing required for ODBC thread-safety standards. By default, the check box is selected.

**Oracle7 only:** When Application Using Threads is enabled, `SQLGetInfo(SQL_ASYNC_MODE)` returns `SQL_AM_NONE`, `SQLSetConnectAttr(SQL_ATTR_ASYNC_ENABLE)` returns "optional feature not implemented," and `SQLSet/GetStmtAttr(SQL_ATTR_ASYNC_ENABLE)` returns "optional feature not implemented." Asynchronous execution is not supported by the Oracle client in a multi-threaded environment.

**Procedure Returns Results:** Select this check box to enable the driver to return result sets from stored procedures/functions. If this check box is selected and you execute a stored procedure that does not return result sets, you will incur a small performance penalty. See "[Stored Procedure Results](#)" on page 199 for details. By default, the check box is not selected.

**Enable Static Cursors for Long Data:** Select this check box to enable the driver to support long columns when using a static cursor. Selecting this check box causes a performance penalty at the time of execution when reading long data. By default, the check box is not selected.

**Use Current Schema for SQLProcedures:** Select this check box to specify only the current user when executing SQLProcedures. When this check box is selected (the default), the call for SQLProcedures is optimized, but only procedures owned by the user are returned.

**Optimize Long Performance (Oracle8 only):** Select this check box to fetch long data directly into the application's buffers rather than allocating buffers and making a copy. This option, when enabled, decreases fetch times on long data; however, it can cause the application to be limited to one active statement per connection. By default, the check box is not selected.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. A logon dialog box is displayed; see "[Connecting to a Data Source Using a Logon Dialog Box](#)" on page 187 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 will display 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].
```

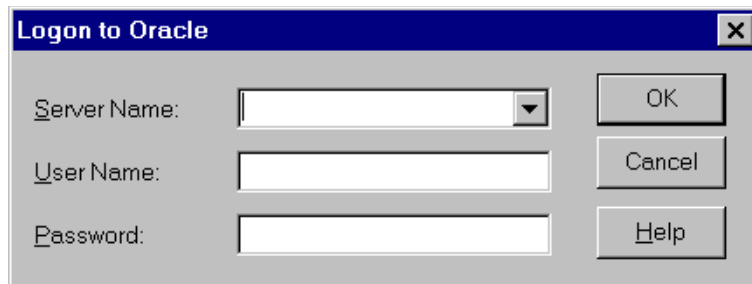
Click **OK**.

- 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.

---

## Connecting to a Data Source 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. For Oracle, the dialog box is as follows:



In this dialog box, perform the following steps:

- 1 Type the client connection string of the computer containing the Oracle database tables you want to access. Or, on Windows, select the string from the Server Name drop-down list box, which displays the names you specified in the ODBC Oracle Driver Setup dialog box.
- 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 system information.

**NOTE:** Oracle has a feature that allows you to connect to Oracle via the operating system user name and password. To connect, use a slash ( / ) for the user name and leave the password blank. To configure the Oracle server/client, refer to the Oracle server documentation.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Oracle is:

```
DSN=Accounting;SRVR=X:QESRVR;UID=JOHN;PWD=XYZZY
```

If the server name contains a semicolon, enclose it in quotation marks:

```
DSN=Accounting;SRVR="X:QE;SRVR";UID=JOHN;PWD=XYZZY
```

[Table 9-1](#) gives the long and short names for each attribute, as well as a description.




To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, "The UNIX Environments," on page 473](#) for information about this file.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 9-1. Oracle Connection String Attributes**

---

Attribute	Description
AppCodePage (ACP) 	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>

**Table 9-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p> <p><b>Oracle7 only:</b> When set to 1, SQLGetInfo(SQL_ASYNC_MODE) returns SQL_AM_NONE, SQLSetConnectAttr(SQL_ATTR_ASYNC_ENABLE) returns "optional feature not implemented," and SQLSet/GetStmtAttr(SQL_ATTR_ASYNC_ENABLE) returns "optional feature not implemented." Asynchronous execution is not supported by the Oracle client in a multi-threaded environment.</p>
ArraySize (AS)	<p>The number of bytes the driver uses for fetching multiple rows. Values can be an integer from 1 up to 4 GB. Larger values increase throughput by reducing the number of times the driver fetches data across the network. Smaller values increase response time, as there is less of a delay waiting for the server to transmit data.</p> <p>The initial default is 60,000.</p>
CatalogOptions (CO)	<p>CatalogOptions={0   1}. Determines whether the result column REMARKS for the catalog functions SQLTables and SQLColumns and COLUMN_DEF for the catalog function SQLColumns have meaning for Oracle. If you want to obtain the actual default value, set CO=1.</p> <p>The initial default is 0.</p>
DataSourceName (DSN)	<p>A string that identifies an Oracle data source configuration in the system information. Examples include "Accounting" or "Oracle-Serv1."</p>

**Table 9-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
EnableDescribeParam (EDP)	<p>EnableDescribeParam={0   1}. Determines whether the ODBC API function SQLDescribeParam is enabled, which results in all parameters being described with a data type of SQL_VARCHAR. This attribute should be set to 1 when using Microsoft Remote Data Objects (RDO) to access data.</p> <p>The initial default is 0.</p>
EnableStaticCursorsForLongData (ESCLD)	<p>EnableStaticCursorsForLongData={0   1}. Determines whether the driver supports long columns when using a static cursor. Using this attribute causes a performance penalty at the time of execution when reading long data.</p> <p>The initial default is 0.</p>
LockTimeOut (LTO)	<p>LockTimeOut={0   -1}. Determines whether Oracle should wait for a lock to be freed before raising an error when processing a Select...For Update statement.</p> <p>When set to 0, Oracle does not wait.</p> <p>When set to -1 (the initial default), Oracle waits forever.</p>
LogonID (UID)	<p>The default logon ID (user name) that the application uses to connect to your Oracle database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. See "<a href="#">Connecting to a Data Source Using a Logon Dialog Box</a>" on page 187 for details.</p>
OptimizeLongPerformance (OLP) <i>Oracle8 Only</i>	<p>OptimizeLongPerformance={0   1}. Determines whether the driver fetches long data directly into the application's buffers rather than allocating buffers and making a copy.</p> <p>When set to 0 (the initial default), the driver allocates buffers for long data.</p> <p>When set to 1, using this setting, the driver fetches long data directly into the application's buffers. The time an application takes to fetch long data decreases; however, it can cause the application to be limited to one active statement per connection.</p>
PacketSize (PS) <i>Oracle7 Only</i>	<p>PacketSize={1024   2048   4096   8192}. Determines the packet size for TCP/IP connections. Any values other than 1024, 2048, 4096, or 8192 are ignored. This value is used only when the ServerName attribute (described later) is set to T for TCP/IP.</p>

**Table 9-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
Password (PWD)	The password that the application uses to connect to your Oracle database. See <a href="#">"Connecting to a Data Source Using a Logon Dialog Box"</a> on page 187 for details.
ProcedureRet Results (PRR)	<p>ProcedureRetResults={0   1}. Determines whether the driver returns result sets from stored procedure functions.</p> <p>When set to 0 (the initial default), the driver does not return result sets from stored procedures.</p> <p>When set to 1, the driver returns result sets from stored procedures. When set to 1 and you execute a stored procedure that does not return result sets, you will incur a small performance penalty. See <a href="#">"Stored Procedure Results"</a> on page 199 for details.</p>
ServerName (SRVR)	<p>The client connection string that designates the server and database to be accessed. The information required varies depending on the client driver that you are using.</p> <p>For Oracle7 remote servers, the SQL*Net 1.x connection string has the following syntax:</p> <pre data-bbox="454 927 922 951"><i>driver_prefix:computer_name[:sid]</i></pre> <p><i>driver_prefix</i> identifies the network protocol being used. The driver prefix can be as follows: P (named pipes), X (SPX), B (NetBIOS), T (TCP/IP), D (DECNet), A (Oracle Async), or AT (AppleTalk). Check your Oracle documentation for other protocols.</p> <p><i>computer_name</i> is the name of the Oracle Listener on your network.</p> <p><i>sid</i> is the Oracle System Identifier and refers to the instance of Oracle running on the host. This item is required when connecting to systems that support more than one instance of an Oracle database.</p>



**Table 9-1. Oracle Connection String Attributes** (cont.)

Attribute	Description
	<p>For local servers, the SQL*Net connection string has the form:</p> <p><i>database_name</i></p> <p><i>database_name</i> identifies your Oracle database.</p> <p>If the SQL*Net connection string contains semicolons, enclose it in quotation marks. See your SQL*Net documentation for more information.</p> <p>For Oracle7 and Oracle8 remote servers, the Oracle TNS Client connection string has the following form:</p> <p><i>TNSNAME</i></p> <p><i>TNSNAME</i> is the alias name of the Oracle Listener on your network.</p>
UseCurrentSchema (UCS)	<p>UseCurrentSchema={0   1}. Determines whether the driver specifies only the current user when executing SQLProcedures. When set to 0, the driver does not specify only the current user. When set to 1 (the initial default), the call for SQLProcedures is optimized, but only procedures owned by the user are returned.</p>

## Data Types

[Table 9-2](#) shows how the Oracle data types are mapped to the standard ODBC data types.

**Table 9-2. Oracle Data Types**

Oracle	ODBC
Char	SQL_CHAR
Date	SQL_TYPE_TIMESTAMP
Long	SQL_LONGVARCHAR

**Table 9-2. Oracle Data Types** (cont.)

Oracle	ODBC
Long Raw	SQL_LONGVARBINARY
Number	SQL_DOUBLE
Number(p,s)	SQL_DECIMAL
Raw	SQL_VARBINARY
Varchar2	SQL_VARCHAR

## Oracle8

Table 9-3 shows how, when connecting to Oracle8 servers, the following additional Oracle8 data types are mapped to the standard ODBC data types. These data types support output parameters to stored procedures.

**Table 9-3. Oracle8 Data Types**

Oracle8	ODBC
Bfile	SQL_LONGVARBINARY*
Blob	SQL_LONGVARBINARY
Clob	SQL_LONGVARCHAR

\* Read-Only

The Oracle8 driver does not support any Abstract Data Types. When the driver encounters an Abstract Data Type during data retrieval, it will return an Unknown Data Type error (SQL State HY000).

---

# Unicode Support

The Oracle drivers determine that the Oracle database is a unicode database if the NLS\_LANG environment variable is set to either:

- AMERICAN\_AMERICA.UTF8
- AMERICAN\_AMERICA.AL24UTFFSS

The Oracle drivers map the Oracle data types as follows:

Oracle Data Type	Mapped to. . .
Char	SQL_WCHAR
Varchar2	SQL_WVARCHAR
Long	SQL_WLONGVARCHAR

CLOB data types are not supported.

These drivers support the Unicode ODBC function calls, called W (Wide) calls (for example, SQLConnectW). These calls are used to accept Unicode datastreams.

## Default Unicode Mapping

The default Unicode mapping for an application's SQL\_C\_WCHAR variable is:

Platform	Default Unicode Mapping
Windows	UCS-2
AIX	UTF-8
HP-UX	UTF-8
Solaris	UTF-8
Linux	UTF-8

## Connection Attributes for Unicode

Two new connection attributes are available to support Unicode. These attributes determine how character data is converted and presented to an application and the database. The connection attributes are:

SQL_ATTR_APP_WCHAR_TYPE (1061)	Sets the SQL_C_WCHAR type for parameter and column binding to the desired unicode type, either SQL_DD_CP_UCS2 or SQL_DD_CP_UTF8. The default is the default Unicode mapping (see the previous section, " <a href="#">Default Unicode Mapping</a> ").
SQL_ATTR_DBMS_CODE_PAGE (1062)	Sets the code page type of the database. The main purpose of this attribute is to set the code page type of a database for drivers that cannot determine the database's code page. The default is SQL_DD_CP_ANSI.

Valid values for the two connection attributes are:

- SQL\_DD\_CP\_ANSI (ANSI code page)
- SQL\_DD\_CP\_UCS2 (UCS-2 code page)
- SQL\_DD\_CP\_UTF8 (UTF-8 code page)

You can set these connection attributes either before or after a connection is made. If the connection attributes are changed after a connection is established, all conversions use the new values.

The driver does not verify that the connection attributes are set only once per connection. Conversions are made based on the current application and database settings.

If the application does **not** set the `SQL_ATTR_DBMS_CODE_PAGE` attribute, the driver tries to determine the database's code page type; if the driver cannot determine the code page type, the driver sets the `SQL_ATTR_DBMS_CODE_PAGE` attribute to `SQL_DD_CP_ANSI`. If the application does set the `SQL_ATTR_DBMS_CODE_PAGE` attribute, the driver, even if it can determine the database's code page type, does not override the value of the attribute set by the application.

If a driver does not support Unicode, `SQLGetConnectAttr` and `SQLSetConnectAttr` return `HYC00`.

These new connection attributes and their valid values can be found in the file `qesqlx.h`, which is installed with the driver.

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on `STATIC` cursors. For example:

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

- 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 regarding Step 1:** 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.

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, `SQL_PERSIST_AS_XML`. A client application must call `SQLSetStmtAttr` with this new attribute as an argument. See the following table for the definition of valid arguments for `SQLSetStmtAttr`.

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

---

## MTS Support

The Oracle8 driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC). Refer to the help file of the "MicroSoft Transaction Server SDK" for details. You must be accessing 8.0.4 or higher servers using Net8 Client 8.0.5.1.0 or higher.

---

## Stored Procedure Results

When the connection option Procedure Returns Results is active, the driver returns result sets from stored procedures/functions. In addition, `SQLGetInfo(SQL_MULT_RESULTS_SETS)` will return "Y" and `SQLGetInfo(SQL_BATCH_SUPPORT)` will return `SQL_BS_SELECT_PROC`. If this option is on and you execute a stored procedure that does not return result sets, you will 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.GTABLE2
    CHARCOL, 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 in the list of procedure arguments. The previously described example would be executed as:

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

where ? is the parameter for the icol argument.

For more information consult your Oracle SQL manual.

---

## Isolation and Lock Levels Supported

Oracle supports isolation level 1 (read committed) and isolation level 3 (serializable isolation—if the server version is Oracle7.3 or higher, or Oracle8.x). Oracle supports record-level locking.

See [Appendix D, "Locking and Isolation Levels,"](#) on page 439 for details.

---

## ODBC Conformance Level

See [Appendix C, "ODBC API and Scalar Functions,"](#) on page 427 for a list of the API functions supported by the Oracle drivers.

The Oracle drivers also support the following functions:

- SQLColumnPrivileges
- SQLDescribeParam (if EnableDescribeParam=1)
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLProcedureColumns



- SQLSetPos
- SQLTablePrivileges

The drivers support the core SQL grammar.

---

## Number of Connections and Statements Supported

The Oracle drivers support multiple connections and multiple statements per connection.



# 10 Connect ODBC for Paradox

Connect ODBC for Paradox (the "Paradox driver") supports Paradox 3x, 4x, 5.x, 7.x, 8.x, and 9.0 tables in the Windows environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows environments supported by this driver.

See the README file shipped with your MERANT DataDirect 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 7, 8, or 9 for Windows

You must have IDAPI32.DLL on your path or in your Windows 9x or Windows Me \SYSTEM directory, or Windows NT or Windows 2000 \SYSTEM32 directory.

## Multiuser Access to Tables

You can query Paradox tables that reside in a shared directory on a network or that are to be shared among applications running on a local workstation. If the tables are on a network server, multiple users can query these tables simultaneously. To share Paradox tables among multiple users, the tables must be located in a shared directory on your network server.

Two connection attributes identify the Paradox database you are accessing: Database (database directory) and NetDir (network directory). The Database setting specifies the directory of Paradox tables that is the database. 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 tables 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 table that another user opens at the same time, the consistency of the data becomes an issue if both individuals are updating the table.

### Locking

The Paradox driver has two locking levels: record locking and table locking. Tables that have no primary key always have a prevent write lock placed on the table when the table is opened.

The table lock is escalated to a write lock when an operation that changes the table is attempted.

Tables 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 tables that are accessed and modified by multiple users.

NOTE: If a table lock is placed on a Paradox table, the Paradox driver prevents users from updating and deleting records but does not prevent them from inserting records into the locked table.

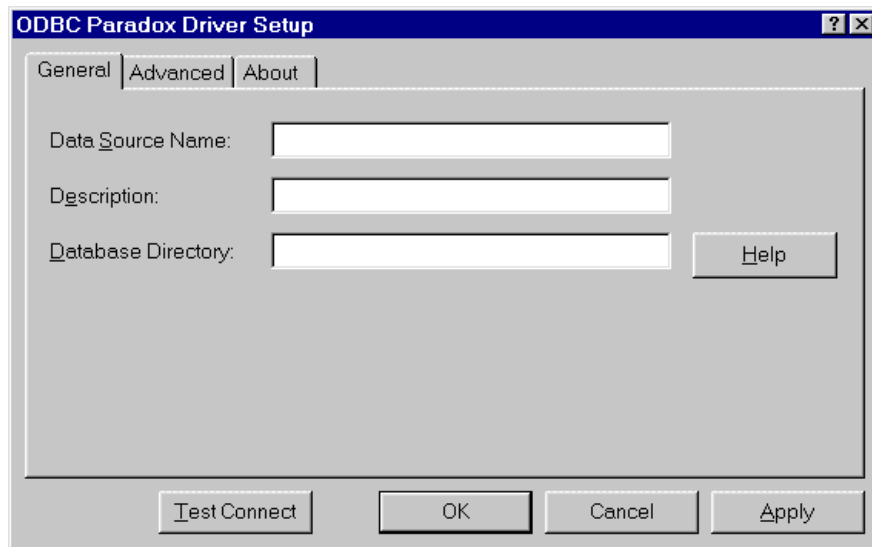
---

## Configuring Data Sources

Data sources are configured and modified through the ODBC Administrator. To configure a Paradox data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Paradox Driver Setup dialog box.

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



**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.

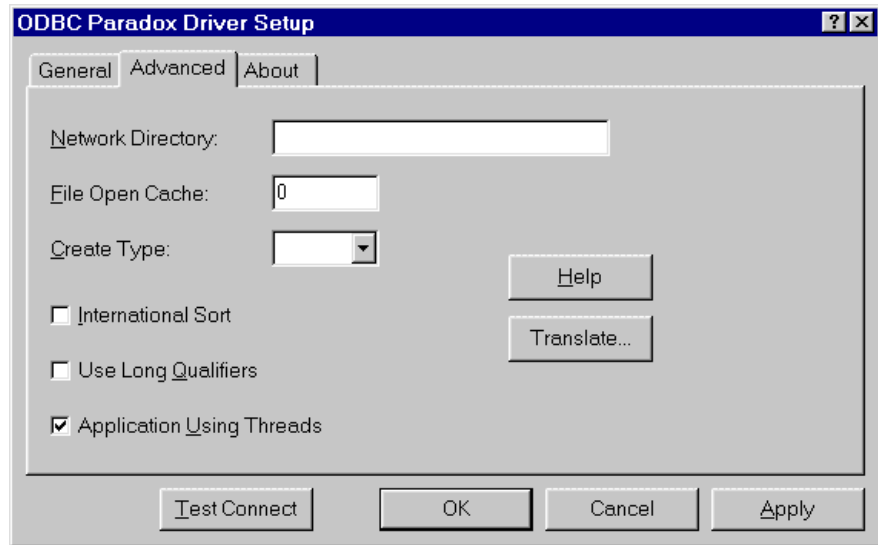
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this Paradox data source configuration in the system information. Examples include "Accounting" or "Paradox-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "Paradox Files on Server number 1."

**Database Directory:** Type the directory in which the Paradox files are stored. If a directory is not specified, the current working directory is used. You can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify ":MYDATA:"

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Network Directory:** Type 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, then every user who accesses it must set this value to point to the same PARADOX.NET file. If not set here, 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.

**File Open Cache:** Type an integer value to specify the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four 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 increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

**Create Type:** Select the Paradox table version used for any Create Table statements that you execute. You can specify the version from the drop-down list or leave the box blank and use the default, which is determined by the Level setting in the Paradox section of the IDAPI configuration file. The numeric values map to the major revision numbers of the Paradox family of products.

If you select Create Type 7, 8, & 9, the Paradox driver supports table names up to 128 characters long. For all other Create Type settings, the driver supports table names up to 8 characters long.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.

**Use Long Qualifiers:** Select this check box to specify whether the driver uses long path names as table qualifiers. If the Use Long Qualifiers check box is selected, path names can be up to 255 characters. The default is unselected (path names can be up to 128 characters).

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You



can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. 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 will display 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**.

- 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.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Paradox is:

```
DSN=PARADOX TABLES;DB=C:\ODBC\EMP;PW=ABC,DEF,GHI
```

[Table 10-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 10-1. Paradox Connection String Attributes**

Attribute	Description
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CreateType (CT)	<p>CreateType={3   4   5   7   8   9   null (blank)}. Specifies the table version for Create Table statements. The numeric values map to the major revision numbers of the Paradox family of products. 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).</p> <p>NOTE: When CreateType is set to 7, 8, or 9, the Paradox driver supports table names up to 128 characters long. For all other CreateType settings, the driver supports table names up to 8 characters long.</p>
Database (DB)	<p>The directory in which the Paradox files are stored.</p> <p>For this attribute, you can also specify aliases that are defined in your IDAPI configuration file, if you have one. To do this, enclose the alias name in colons. For example, to use the alias MYDATA, specify "Database=:MYDATA:"</p>
DataSourceName (DSN)	<p>A string that identifies a Paradox data source configuration in the system information. Examples include "Accounting" or "Paradox-Serv1."</p>

**Table 10-1. Paradox Connection String Attributes** (cont.)

Attribute	Description
DeferQueryEvaluation (DQ)	<p data-bbox="486 317 1236 404">DeferQueryEvaluations={0   1}. Determines when a query is evaluated—after all records are read or each time a record is fetched.</p> <p data-bbox="486 423 1236 708">When set to 0, the driver generates a result set when the first record is fetched. The driver reads all records, evaluates each one against the Where clause, and compiles a result set containing the records that satisfy the search criteria. This process slows performance when the first record is fetched, but activity performed on the result set after this point is much faster, because the result set has already been created. You do not see any additions, deletions, or changes in the database that occur while working from this result set.</p> <p data-bbox="486 727 1236 1038">When set to 1 (the initial default), the driver evaluates the query each time a record is fetched, and stops reading through the records when it finds one that matches the search criteria. This setting avoids the slowdown while fetching the first record, but each fetch takes longer because of the evaluation taking place. The data you retrieve reflect the latest changes to the database; however, a result set is still generated if the query is a Union of multiple Select statements, if it contains the Distinct keyword, or if it has an Order By or Group By clause.</p>
FileOpenCache (FOC)	<p data-bbox="486 1057 1236 1341">The maximum number of unused table opens to cache. For example, when FileOpenCache=4, and a user opens and closes four 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 using file open caching is increased performance. The disadvantage is that a user who tries to open the table exclusively may get a locking conflict even though no one appears to have the table open.</p> <p data-bbox="486 1361 1236 1381">The initial default is 0, which means no file open caching.</p>

**Table 10-1. Paradox Connection String Attributes** (cont.)

Attribute	Description
IntlSort (IS)	<p data-bbox="525 314 1272 404">IntlSort={0   1}. Determines the order in which records are retrieved when you issue a Select statement with an Order By clause.</p> <p data-bbox="525 421 1272 548">When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p data-bbox="525 565 1272 748">When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>
NetDir (ND)	<p data-bbox="525 765 1272 1017">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, then every user who accesses it must set this value to point to the same PARADOX.NET file. If 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.</p>
Passwords (PW)	<p data-bbox="525 1034 1272 1157">A password or list of passwords. You can add 1 to 50 passwords into the system using a comma-separated list of passwords. Passwords are case-sensitive. For example, Passwords=psw1, psw2, psw3.</p>

**Table 10-1. Paradox Connection String Attributes** (cont.)

Attribute	Description
UltraSafeCommit (USF)	<p data-bbox="486 314 1240 378">UltraSafeCommit={0   1}. Determines when the driver flushes its changes to disk.</p> <p data-bbox="486 392 1240 517">When set to 0, (the initial default), the driver flushes its changes to disk when the table is closed or when internal buffers are full. In this case, a machine "crash" before closing a table may cause recent changes to be lost.</p> <p data-bbox="486 531 1240 586">When set to 1, the driver flushes its changes to disk at each COMMIT. This decreases performance.</p>
UseLongQualifiers (ULQ)	<p data-bbox="486 604 1240 668">UseLongQualifiers={0   1}. Determines whether the driver uses long path names as table qualifiers.</p> <p data-bbox="486 682 1240 772">When set to 0 (the initial default), the driver does not use long path names as table qualifiers (the maximum path name length is 128 characters).</p> <p data-bbox="486 786 1240 843">When set to 1, the driver uses long path names (the maximum path name length is 255 characters).</p>

---

## Data Types

Table 10-2 shows how the Paradox data types are mapped to the standard ODBC data types. It also identifies the types supported by Paradox versions 3.x, 4.x, and 5.x and higher. These Paradox data types can be used in a Create Table statement. See [“Create Table Statement” on page 217](#) for the syntax of the Create Table statement.

**Table 10-2. Paradox Data Types**

<b>Paradox</b>	<b>ODBC</b>	<b>3.x Support</b>	<b>4.x Support</b>	<b>5.x and Higher Support</b>
Alpha	SQL_CHAR	Yes	Yes	Yes
AutoIncrement	SQL_INTEGER	No	No	Yes
BCD	SQL_DECIMAL	No	No	Yes
Binary*	SQL_LONGVARBINARY	No	Yes	Yes
Bytes*	SQL_BINARY	No	No	Yes
Date	SQL_TYPE_DATE	Yes	Yes	Yes
Formatted Memo*	SQL_LONGVARBINARY	No	Yes	Yes
Graphic*	SQL_LONGVARBINARY	No	Yes	Yes
Logical*	SQL_BIT	No	No	Yes
Long Integer	SQL_INTEGER	No	No	Yes
Memo*	SQL_LONGVARCHAR	No	Yes	Yes
Money	SQL_DOUBLE	Yes	Yes	Yes
Number	SQL_DOUBLE	Yes	Yes	Yes
OLE*	SQL_LONGVARBINARY	No	Yes	Yes
Short	SQL_SMALLINT	Yes	No	Yes
Time	SQL_TYPE_TIME	No	No	Yes
TimeStamp	SQL_TYPE_TIMESTAMP	No	No	Yes

\* Cannot be used in an index. Of these types, only Logical can be used in a Where clause.

---

## 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 [Appendix A, “SQL for Flat-File Drivers” on page 393](#) 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 table or to delete (drop) a single column.

The Alter Table statement has the form:

```
ALTER TABLE table_name {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]}
```

*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 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 system information. 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 tables support "not null" columns. In the cases where "not null" is not supported, this restriction is ignored and the driver returns a warning if "not null" is specified for a column. The "not null" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not null" restriction).

A sample Create Table statement to create an employee database table is:

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL 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 up to 50 passwords.

Paradox tables 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 tables 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 table), 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 tables. 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 up to 15 characters in length, including blanks. If your password includes lower-case letters or nonalphanumeric characters, enclose it in single quotation marks (').

## Encrypting a Paradox Table

The Encrypt statement associates a password with a table. For example:

```
ENCRYPT emp USING PASSWORD test
```

## Accessing an Encrypted Paradox Table

To access an encrypted Paradox table, 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 table. 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 table.

## Decrypting a Paradox Table

The Decrypt statement disassociates a password from a table. You no longer need to enter the password to open the table. 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 table, you must continue entering the password to open the table.

---

## Index Files

An index is used to read records in sorted order and to improve performance when selecting records and joining tables. 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 table. The primary key fields of a primary index consist of one or more consecutive fields in the table, beginning with the first field in the table. A table can have only one primary index.

Collectively, the primary key fields uniquely identify each record in the Paradox table. Thus, no two records in a Paradox table can share the same values in their primary key fields. Once a primary index is created for a Paradox table, the table'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 table, the primary index is updated automatically to reflect these changes.

A primary index is a single file with the same name as the table 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, and 9 tables support UNIQUE secondary indexes. Refer to [“Create and Drop Index Statements” on page 223](#) for more information.

A non-primary index is defined by specifying one or more fields in the Paradox table that constitute the non-primary key field. It allows Paradox to sort each record in the table 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 table 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 table before you create a maintained, non-primary index.

The Paradox driver uses non-maintained indexes only for read-only queries on locked tables. 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 table on which the non-primary index is based; one of these files has an .Xnn extension while the other has a .Ynn 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, and 9 single-field non-primary indexes that are case-sensitive have the same name as their associated

table 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 table 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
 (column [,column...])
```

The UNIQUE keyword is optional; the index is unique whether or not you include this keyword.

*table\_name* is the name of the table 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 table, beginning with the first field in the table.

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 3.0, 3.5, 4.0, 4.5, and 5.0 tables, the syntax for creating a non-primary index is:

```
CREATE INDEX index_name
[/NON_MAINTAINED] [/CASE_INSENSITIVE] ON
 (column [, column...])
```

For Paradox 7, 8, and 9 tables, the syntax for creating a non-primary index is:

```
CREATE [UNIQUE] INDEX index_name
[/NON_MAINTAINED] [/CASE_INSENSITIVE] ON
 [column [DESC] [, column...]
```

For Paradox 7, 8, and 9 tables only (when the Create Type is 7, 8, or 9), 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.

*table\_name* is the name of the table on which the index is to be based.

*column* is the name of a column that is included as a the key field for the index.



For Paradox 7, 8, or 9 tables only (when the Create Type is 7, 8, or 9), the DESC keyword creates a non-primary index that uses descending keys.

Paradox 3.0 and 3.5 tables cannot have composite or case-insensitive indexes. When you create a non-primary index for Paradox 3.0 and 3.5 tables, follow these rules:

- Specify only one column name.
- Do not use the CASE\_INSENSITIVE switch.
- Use the column name as the index name.

For example:

```
CREATE INDEX last_name ON emp (last_name)
```

## 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 table from which the index is being dropped. The pathname can be either the fully qualified pathname or, if the table is specified with the Database attribute of the connection string, a simple table 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). Use the column name as the index name when dropping indexes from Paradox 3.5 or 3.0 tables.

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 don't use transactions, the Paradox driver immediately executes Insert, Update, and Delete statements on the database tables 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. See [Appendix D, "Locking and Isolation Levels" on page 439](#) for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the Paradox driver. In addition, the following functions are supported:

- SQLSetPos
- SQLPrimaryKeys

When used with Paradox 5 or Paradox 7, 8, and 9 tables, 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.

---

## Number of Connections and Statements Supported

The Paradox database system supports multiple connections and multiple statements per connection.



# 11 Connect ODBC for PROGRESS

Connect ODBC for PROGRESS (the "PROGRESS driver") supports database version 7.3 and version 8.x of the PROGRESS database system in the Windows and UNIX environments.

The PROGRESS driver runs in the Windows environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the PROGRESS driver.

---

## Driver Requirements

To access a PROGRESS database with a PROGRESS driver, your system must include the following items:

- On the client computer: PROGRESS Version 8.2A installation of Client Networking for Windows NT.
- On the Server computer:
  - PROGRESS Version 7.3C (or later) Server Networking and Database Server installed on the platform you choose as your database server machine.
  - PROGRESS Version 7.3C (or later) Client Networking installed on the platform you choose for your OID, if you choose to connect via server rather than directly.

NOTE: The PROGRESS driver accesses PROGRESS databases that are created with Version 7.3C or later. This driver cannot access Version 6 PROGRESS databases.

Before using the driver, you must set the IDLC environment variable to your PROGRESS DLC directory. For example:

```
set IDLC=C:\DLC
```

---

## Configuring Data Sources

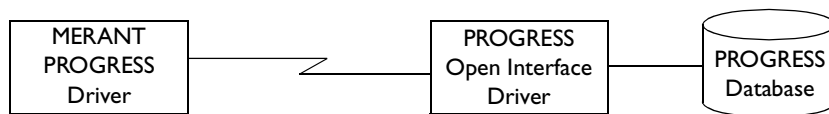
The Connect ODBC PROGRESS driver supports the following data source configurations:

- Remote OID with direct database access
- Remote OID with database access via server

The PROGRESS driver does not support remote, single-user configurations.

### Remote OID with Direct Access

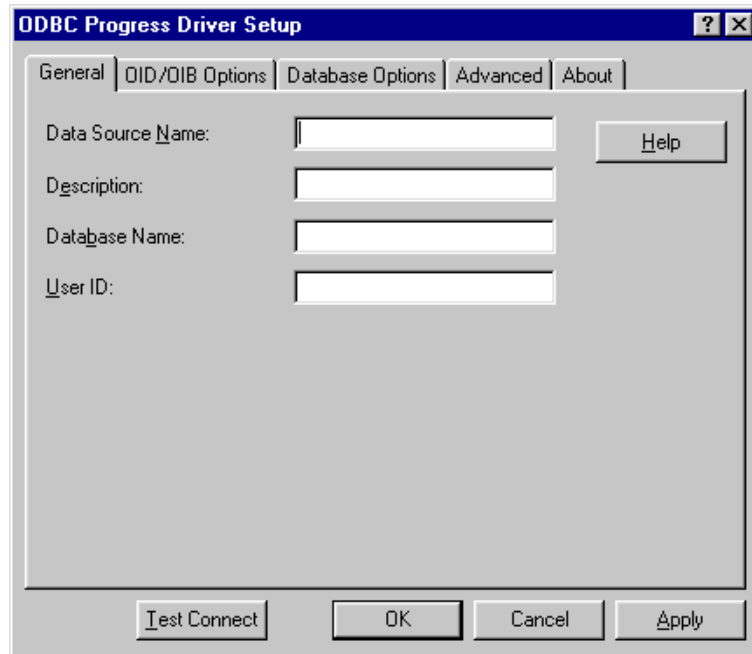
The following figure shows a Direct Access configuration.



**To configure a PROGRESS data source for a remote OID with direct database access (no server process) configuration:**

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Progress Driver Setup dialog box.

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



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

**Data Source Name:** Type a string that identifies this PROGRESS data source configuration in the system information. Examples include "Accounting" or "PROG-Serv1."

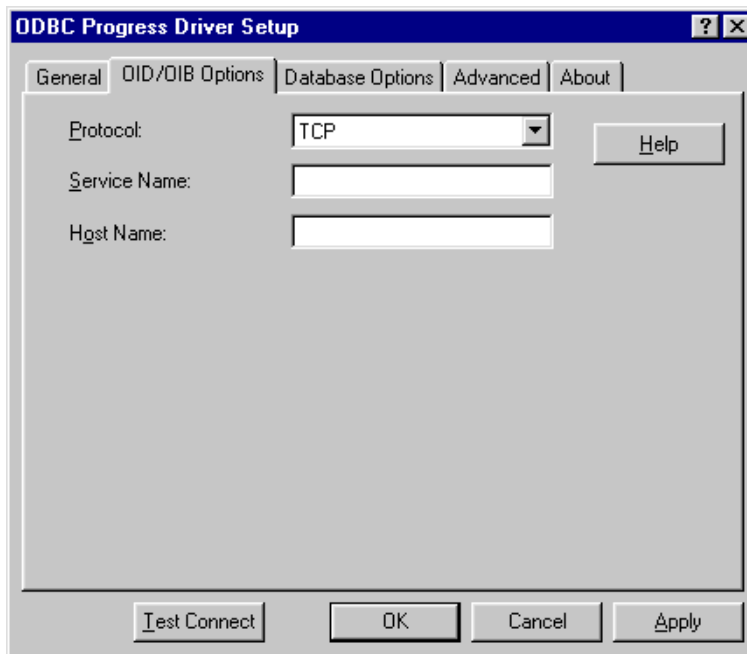
**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "PROGRESS on Server number 1."

**Database Name:** Type the name of the database to which you want to connect by default.

**User ID:** Type the default logon ID used to connect to your PROGRESS database. Your ODBC application may override

this value or you may override this value in the Logon dialog box or connection string.

- 4 Click the **OID/OIB Options** tab to specify OID options.



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

**Protocol:** Select TCP, NETBIOS, or SPX.

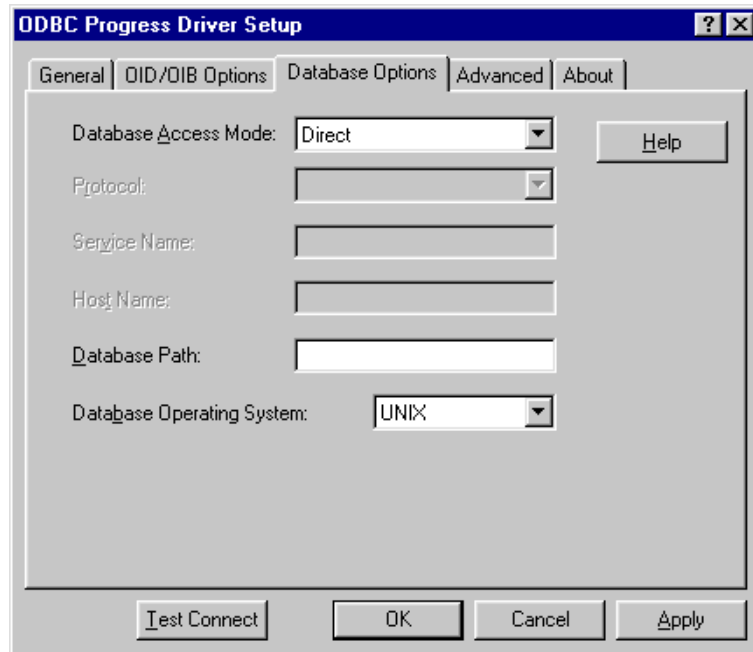
**Service Name:** Type the service name of the OID as listed in the services file.

**Host Name:** Type the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.



- 5 Click the **Database Options** tab to specify database options.



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

**Database Access Mode:** Select Direct.

**Protocol:** Not applicable for this configuration.

**Service Name:** Not applicable for this configuration.

**Host Name:** Not applicable for this configuration.

**Database Path:** Type the fully qualified directory path on the server containing the database (not including the database name or extension).

**Database Operating System:** Select the operating system under which the database is stored. This value determines which qualifier separator to use when building the database path: backslash (\) for Windows or slash (/) for UNIX.

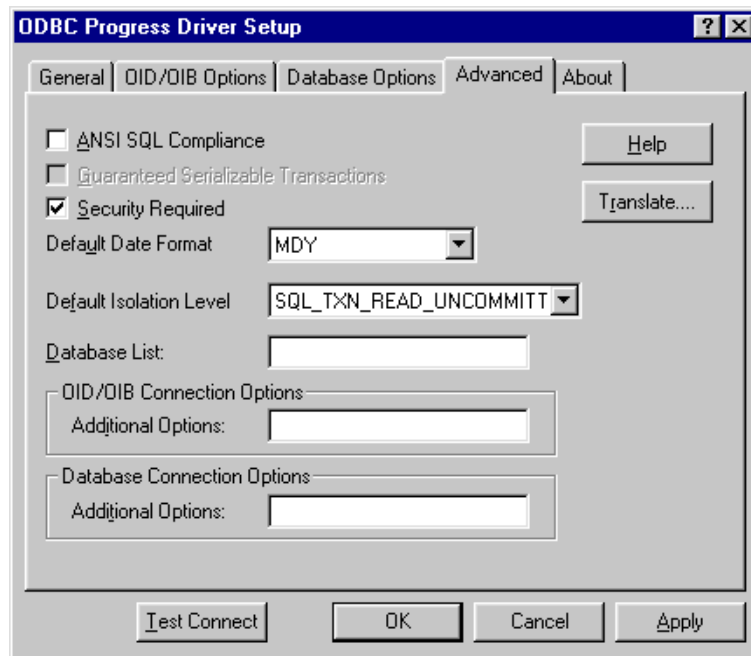
If you select Ignore, you must type a qualified separator as the last character in the Database Path field (described above).

For example, for Windows NT:

c:\progress\  
 and for UNIX:

/databases/progress/

- 6 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

**ANSI SQL Compliance:** Select this check box to allow case-sensitive character data.

**Guaranteed Serializable Transactions:** Select this check box to allow serializable transactions. To select this option, the ANSI SQL Compliance check box must be selected.

**Security Required:** Select this check box to determine whether the logon ID (UID) is required in the connection string.

**Default Date Format:** Select MDY, DMY, or YMD as the default date format.

**Default Isolation Level:** Select either SQL\_TXN\_READ\_COMMITTED or SQL\_TXN\_READ\_UNCOMMITTED as the default isolation level for concurrent transactions. You must select SQL\_TXN\_READ\_UNCOMMITTED to work around specific locking problems.

**Database List:** Type a comma-delimited list of up to four databases to which the driver will connect in addition to the default database. The default database must not be part of this list. The same User ID, password, and path as the default database must be applicable to these databases. In a session, users can access only the default database and those databases specified in this list.

**OID/OIB Connection Options/Additional Options:** Type additional PROGRESS parameters for Open Interface Driver/Open Interface Broker installation. The format for specifying these options is:

*-parameter [argument]...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Database Connection Options/Additional Options:** Type additional PROGRESS parameters for the database connection. The format for specifying these options is:

*-parameter [argument]...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 7 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 is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box" on page 247](#) 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 will display an appropriate error message.

Verify that all required client software is properly installed. If it is not, you will see the message:

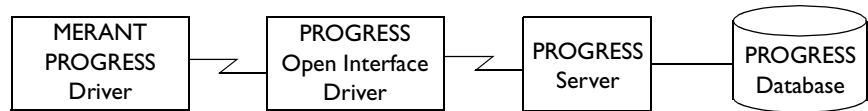
```
Specified driver could not be loaded due to system  
error [xxx].
```

Click **OK**.

- 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.

## Remote OID with Database Access via Server

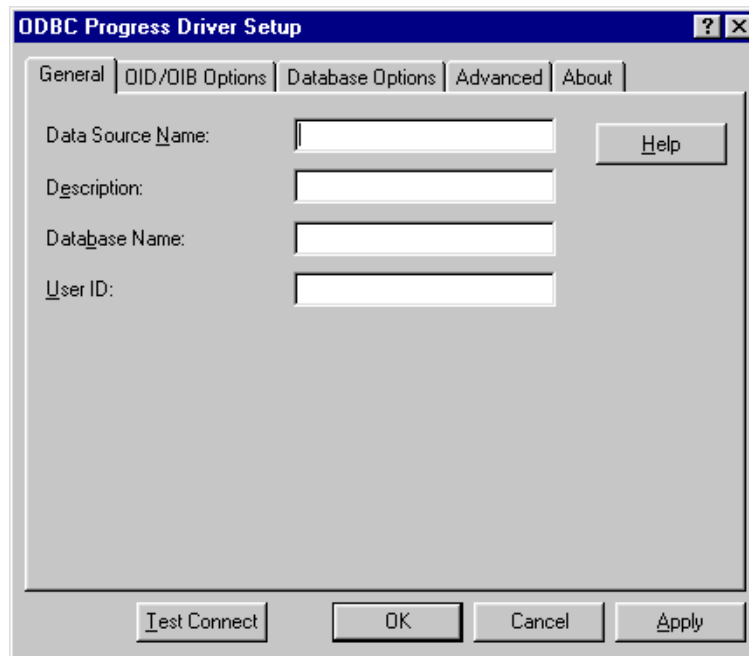
The following figure shows a database access via server configuration.



**To configure a PROGRESS data source for a remote OID with database access via a server:**

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Progress Driver Setup dialog box.

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



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

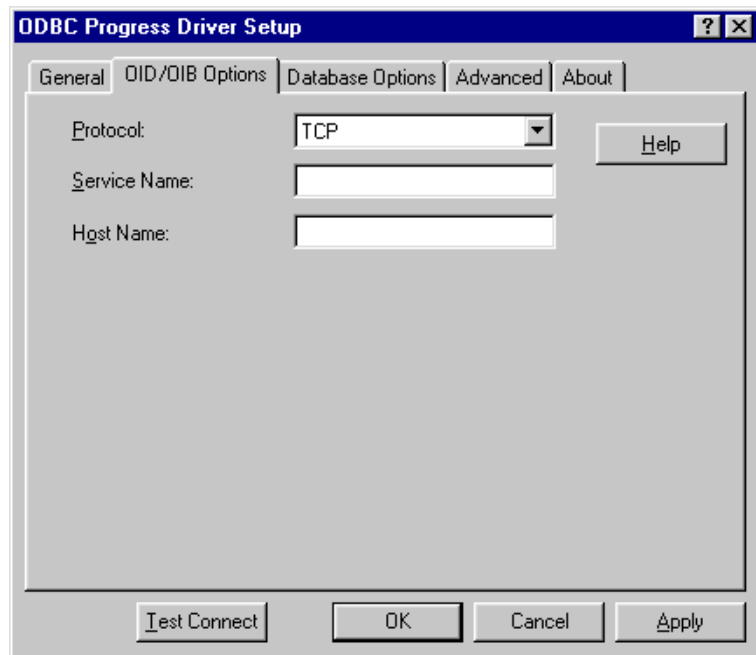
**Data Source Name:** Type a string that identifies this PROGRESS data source configuration in the system information. Examples include "Accounting" or "PROG-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "PROGRESS on Server number 1."

**Database Name:** Type the name of the database to which you want to connect by default.

**User ID:** Type the default logon ID used to connect to your PROGRESS database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

- 4 Click the **OID/OIB Options** tab to specify OID options.



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

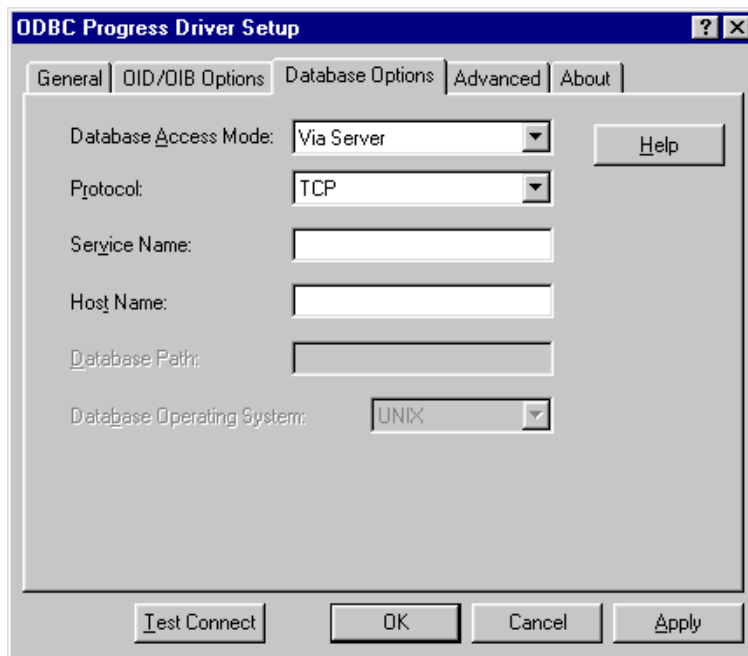
**Protocol:** Select TCP, NETBIOS, or SPX.

**Service Name:** Type the service name of the OID as listed in the services file.

**Host Name:** Type the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.

- 5 Click the **Database Options** tab to specify database options.



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

**Database Access Mode:** Select Via Server.

**Protocol:** Select the type of protocol your configuration uses: NETBIOS, SPX, or TCP.

**Service Name:** Type the name of the database service as listed in the services file.

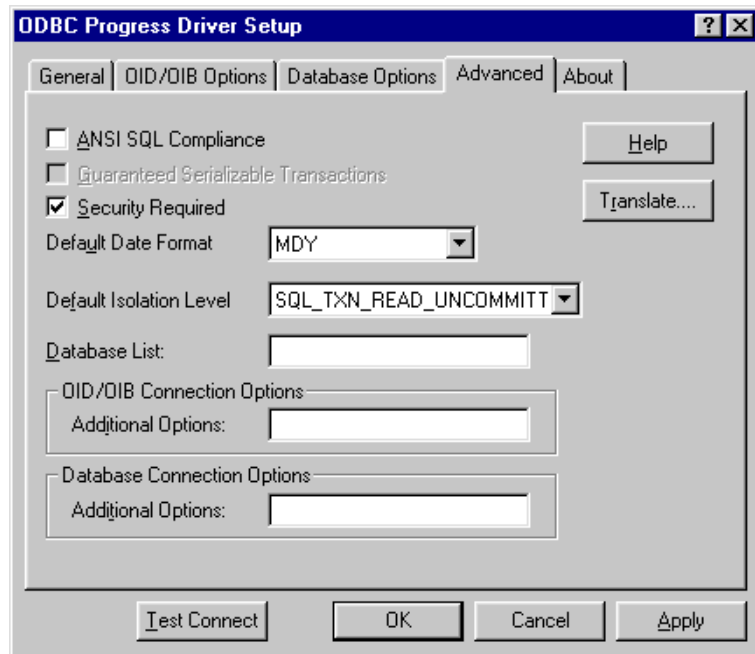
**Host Name:** Type the host name of the server process machine.

**Database Path:** Not applicable for this configuration.

**Database Operating System:** Not applicable for this configuration.



- 6 Optionally, click the **Advanced** tab to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

**ANSI SQL Compliance:** Select this check box to allow case-sensitive character data.

**Guaranteed Serializable Transactions:** Select this check box to allow serializable transactions. To select this option, the ANSI SQL Compliance check box must be selected.

**Security Required:** Select this check box to determine whether the logon ID (UID) is required in the connection string.

**Default Date Format:** Select MDY, DMY, or YMD as the default date format.

**Default Isolation Level:** Select either SQL\_TXN\_READ\_COMMITTED or SQL\_TXN\_READ\_UNCOMMITTED as the default isolation level for concurrent transactions. You must select SQL\_TXN\_READ\_UNCOMMITTED to work around specific locking problems.

**Database List:** Not applicable for this configuration.

**OID/OIB Connection Options/Additional Options:** Type additional PROGRESS parameters for Open Interface Driver/Open Interface Broker installation. The format for specifying these options is:

*-parameter [argument]...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Database Connection Options/Additional Options:** Type additional PROGRESS parameters for the database connection. The format for specifying these options is:

*-parameter [argument]...*

For more information about these parameters, refer to the system administration documentation for PROGRESS.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 7 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 is displayed; see ["Connecting to a](#)

[Data Source Using a Logon Dialog Box](#) on page 247 for details.

- 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 will display an appropriate error message.

Verify that the 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**.

- 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.

---

## Configuring the Progress Environment

To configure the Progress environment for access to a Progress database, you must:

- 1 Set the appropriate system variables. See ["Setting System Variables"](#) for details.
- 2 Start Open Interface Broker in the directory containing the database to ensure a proper self-serving OID client:

```
%DLC%\BIN\oibrkr32.exe -SV -S pdro82oid
```

where "pdro82oid" is the service name as specified in the services file.

### 3 Start the database server:

```
%DLC%\BIN\_mprosrv database_name
```

The following is sample connection information:

```
Database=test  
UID=<blank>  
PWD=<blank>  
OID/OIB Protocol=TCP  
OID/OIB Service Name=pdro82oid  
OID/OIB Host Name=localhost  
Database Access Mode=Direct  
Database Path=c:\protmp82  
Database Operating System=Windows
```

## Setting System Variables

The following system variables must be set on your client and server machines. The client and server machine can be the same machine. On a Windows 9x machine, you can add the variables in your autoexec.bat using the syntax:

```
"SET DLC=c:\dlc"
```

On a Windows NT or Windows 2000 machine, change the settings in your Control Panel using the System Icon.

On a UNIX server machine, set the variables using the syntax:

```
"DLC=/hd130/pro80/dlc ; export DLC"
```

On UNIX the variable names and values are case sensitive.

**On the client machine, set the following variables:**

<b>Variable</b>	<b>Value</b>	<b>Comment</b>
DLC	c:\dlc	The directory where you installed the Progress client pieces
IDLC	c:\dlc	Required only for 8.1 or higher clients
PROMSGS	c:\dlc\promsgs	The directory and name of the message file
IPROMSGS	c:\dlc\promsgs	Required only for 8.1 or higher clients
PATH	c:\dlc\bin;c:\dlc;c:\winNT\system32;%path%	Example for Windows NT
TEMP	c:\temp	Specify an existing directory where you have full privileges (write, delete, etc.)

**On a Windows NT server machine, set the following variables:**

<b>Variable</b>	<b>Value</b>	<b>Comment</b>
DLC	c:\dlc	The directory where you installed the Progress server pieces
PROOIBRK	%dlc%\bin\oibrkr32.exe	The directory and name of the Progress Broker executable
PROOIDRV	%dlc%\bin\oidrvr32.exe	The directory and name of the Progress OID executable
PATH	c:\dlc\bin;%path%	

**NOTE:** If you have the client and server pieces on the same machine, you must set all of the client machine variables and Windows NT server machine variables.

**On a UNIX server machine, set the following variables:**

<b>Variable</b>	<b>Value</b>	<b>Comment</b>
DLC	hd130/dlc	The directory where you installed the Progress server pieces
PROOIBRK	\$dlc/bin/_prooibk	The directory and name of the Progress Broker executable
PROOIDRV	\$dlc/bin/_prooidv	The directory and name of the Progress OID executable
PATH	/hd130/dlc/bin:\$PATH	

**NOTE:** On a UNIX server, the variables and values are case-sensitive.

## Services Files

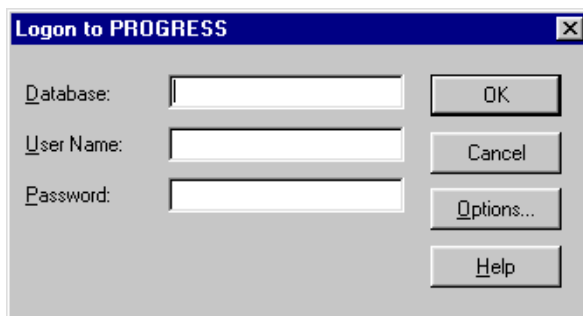
The name of the Progress database service and broker service must be set in the services file on both the client and service machines. Be sure that the port you assign to the service is not in use by another service and restrict the name to a maximum of eight alphanumeric characters, dashes are not valid characters.

**NOTE:** Entries in the services file are case-sensitive.

---

## Connecting to a Data Source 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. For PROGRESS, the dialog box is as follows:



In this dialog box, perform the following steps:

- 1 Type the name of the database to which you want to connect (without a path or an extension).
- 2 If required, type your user name.
- 3 If required, type your password.

User name and password are optional parameters. You can log on to the database without these parameters, but you may not be able to create, delete, or manipulate the data.

- 4 Click **Options** to display the PROGRESS Logon Options dialog box.

The screenshot shows a dialog box titled "PROGRESS Logon Options". It contains two main sections of configuration options:

- OID/OIB Connection Options:**
  - Protocol: A dropdown menu.
  - Service Name: A text input field.
  - Host Name: A text input field.
- Database Connection Options:**
  - Database Access Mode: A dropdown menu.
  - Protocol: A dropdown menu with "TCP" selected.
  - Service Name: A text input field.
  - Host Name: A text input field.
  - Database Path: A text input field.
  - Database Operating System: A dropdown menu.

On the right side of the dialog, there are three buttons: "OK", "Cancel", and "Help".

Provide any of the following information; then, click **OK** to complete the logon and to update the values in the system information.

**Protocol:** Select the protocol used for communication with the OID; TCP, NETBIOS, or SPX.

**Service Name:** Type the service name of the OID as listed in the services file.

**Host Name:** Type the host name of the OID/OIB machine.

An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.

**Database Access Mode:** Select how the OID accesses the database. Select Direct if the OID access the database file directly. Select Via Server if the OID accesses the database through a server process.



**Protocol:** Select the type of protocol your configuration uses: NETBIOS, SPX, or TCP. This option is available only when the Database Access option is set to Via Server.

**Service Name:** For Via Server configurations, type the service name of the server process. Not applicable for Direct configurations.

**Host Name:** For Via Server configurations, type the host name of the server process machine. Not applicable for Direct configurations.

**Database Path:** Type the path name of the database when the OID accesses the database directly (not via a server). This setting is necessary only when the Database Connection Options/Database Access option is set to Direct.

**Database Operating System:** Select the operating system under which the database is stored or select Ignore. This value determines which qualifier separator to use when building the database path: backslash (\) for Windows or slash (/) for UNIX. This setting is necessary only if the Database Connection Options/Database Access is set to Direct.

**NOTE:** If you select Ignore from the drop-down list, you must type a qualifier separator as the last character in the Database Path field (described above). For example, for Windows NT:

`c:\progress\`

and for UNIX:

`/databases/progress/`

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for PROGRESS is:

```
DSN=PROGRESS;DB=PAYROLL;UID=JOHN;PWD=XYZZY;  
OIDP=TCP;OIDS=PRO81OID;OIDH=PROSRV1; DBAM=VIASERVER;  
DBPR=TCP;DBS=PRO81SRVPAYROLL; DBH=PROSRV1;ASC=0
```

[Table 11-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 11-1. PROGRESS Connection String Attributes**

Attribute	Description
AnsiSQL Compliance (ASC)	<p>AnsiSQLCompliance={0   1}. Determines whether the values in a character column, and comparisons made to them, must be case-sensitive.</p> <p>When set to 0 (the initial default), comparisons are not case-sensitive.</p> <p>When set to 1, comparisons are case-sensitive.</p>
Database (DB)	The name of the database to which you want to connect.
DatabaseOS (DBOS)	DatabaseOS={Windows   UNIX   Ignore   VMS}. Specifies the operating system under which the database is stored. This value determines which qualifier separator to use when building the database path: backslash (\) for Windows or slash (/) for UNIX.
DataSourceName (DSN)	A string that identifies a PROGRESS data source configuration in the system information. Examples include "Accounting" or "PROG-Serv1."
DBAccessMode (DBAM)	<p>DBAccessMode={Direct   Via Server}. Specifies how the OID accesses the database.</p> <p>When set to Direct, the OID accesses the database file directly.</p> <p>When set to Via Server, the OID accesses the database through a server process.</p>
DBHost (DBH)	When DBAccessMode=Via Server, specify the host name of the server machine using this attribute.
DBOptions (DBO)	<p>A string that contains parameters to be used when the OID connects to the specified database(s). The format for specifying OID connection options is:</p> <p><i>DBOptions=-syntax parameter_definition. . .</i></p>
DBPath (DBPA)	When DBAccessMode=Via Server, specify the path name of the database (without the database name) using this attribute.
DBProtocol (DBPR)	DBProtocol={NETBIOS   SPX   TCP}. When DBAccessMode=Via Server, specify the protocol to connect the Open Interface Driver (OID) to the PROGRESS database server using this attribute.
DBService (DBS)	When DBAccessMode=Via Server, specify the name of the database service listed in the services files.

**Table 11-1. PROGRESS Connection String Attributes** (cont.)

DefaultDate Format (DDF)	DefaultDateFormat={MDY   DMY   YMD}. Determines whether PROGRESS uses MDY, DMY, or YMD as the default date format.
DefaultIsolation Level (DIL)	DefaultIsolationLevel={0   1}. Specifies the default isolation level for concurrent transactions.  When set to 0, the default isolation level is SQL_TXN_READ_COMMITTED.  When set to 1 (the initial default), the default isolation level is SQL_TXN_READ_UNCOMMITTED. You should use this value to work around specific locking problems.
GSTransaction (GST)	GSTransaction={0   1}. Determines whether serializable transactions are allowed.  When set to 0 (the initial default), serializable transactions are not allowed.  When set to 1, serializable transactions are allowed. This attribute can be set to 1 only if the AnsiSQLCompliance attribute is set to 1.
LogonID (UID)	The default logon ID (user name) used to connect to your PROGRESS database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID. This ID is case-sensitive.
OIDHost (OIDH)	The host name of the Open Interface Driver (OID). (If the OID is local, enter the local machine's host name in this field).  An Open Interface Broker or Open Interface Driver must be running on the specified host to connect to the database.
OIDOptions (OIDO)	A string that contains parameters to be passed to the OID when it is auto-started. The format for specifying OID connection options is:  <code>OIDOptions=-syntax parameter_definition. . .</code>
OIDProtocol (OIDP)	OIDProtocol={TCP   NETBIOS   SPX}. Specifies the protocol to connect the PROGRESS driver to the Open Interface Driver (OID).
OIDService (OIDS)	The service name of the Open Interface Driver (OID).
Password (PWD)	A case-sensitive password.
SecurityRequired (SR)	SecurityRequired={0   1}. Determines whether the logon ID (UID) is required in the connection string.  The initial default is 1; the UID is required.

---

# Data Types

[Table 11-2](#) shows how the PROGRESS data types are mapped to the standard ODBC data types.

---

**Table 11-2. PROGRESS Data Types**

---

<b>PROGRESS</b>	<b>ODBC</b>
Character	SQL_VARCHAR
Date	SQL_TYPE_DATE
Decimal	SQL_DECIMAL
Float	SQL_FLOAT
Integer	SQL_INTEGER
Logical	SQL_BIT

---

## Isolation and Lock Levels Supported

PROGRESS supports isolation level 1 (read committed) if the data source was defined without Guaranteed Serializable Transactions; otherwise, PROGRESS supports isolation level 3 (serializable). PROGRESS supports record-level locking. See [Appendix D, "Locking and Isolation Levels" on page 439](#) for details.

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the functions supported by the PROGRESS driver. The driver supports the core SQL grammar.

The driver also supports the function `SQLSetPos` and forward and backward scrolling with `SQLExtendedFetch` and `SQLFetchScroll`.

---

## Number of Connections and Statements Supported

The PROGRESS database system supports multiple connections and multiple statements per connection.

# 12 Connect ODBC for SQL Server

Connect ODBC for SQL Server (the "SQL Server driver") supports the SQL Server database system versions 6.5 and 7.0, and SQL Server 2000 database system available from Microsoft in the Windows and UNIX environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the SQL Server driver.

---

## Driver Requirements

This section provides the driver requirements for using the SQL Server driver on both Windows and UNIX platforms.



### Windows

The SQL Server version 7.0 ODBC driver requires the SQL Server 7.0 versions of Net-Library DLL files, which are installed when you install Connect ODBC for SQL Server. The driver communicates with network software through the SQL Server Net-Library interface.

The SQL Server driver requires Microsoft client software; it does *not* work with Sybase System 10 or 11 software.



## UNIX

To use the SQL Server driver on UNIX, you must have TCP/IP configured on both the UNIX client and the Windows NT server on which the SQL Server database resides. The UNIX SQL Server TCP/IP network client library is built into the UNIX SQL Server ODBC driver. Nothing else is required beyond the normal ODBC environment.

The SQL Server Client configuration has been merged with the ODBC driver configuration and is set in the system information file.

---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.



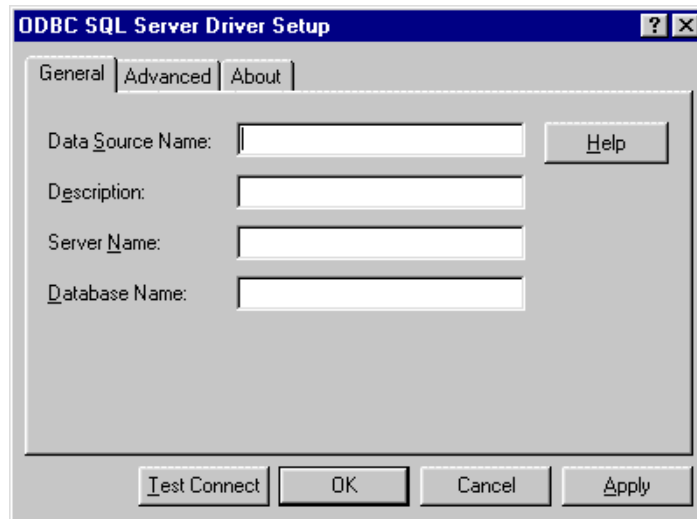
In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 12-2 on page 269](#). See [Appendix H, "The UNIX Environments" on page 473](#) for information about editing the file.

### To configure an SQL Server data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select SQL Server and click **Finish** to display the ODBC SQL Server Driver Setup dialog box.

If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC SQL Server Driver Setup dialog box.





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

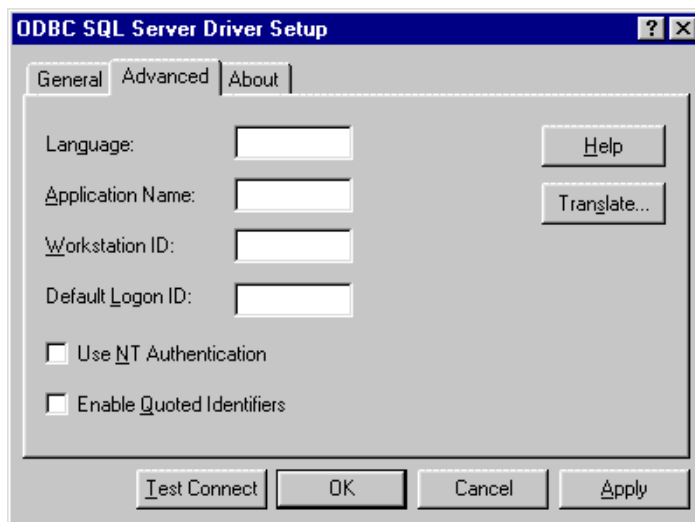
**Data Source Name:** Type a string that identifies this SQL Server data source configuration in the system information. Examples include "Accounting" or "SQL Server-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "SQL Server on Server number 1."

**Server Name:** Type the name of the server that contains the database you want.

**Database Name:** Type the name of the database to which you want to connect by default. If you do not specify a value, the default database defined by SQL Server is used.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Language:** Type the national language to be used by the client. The default is English.

**Application Name:** Type the name SQL Server uses to identify your application.

**Workstation ID:** Type the workstation ID used by the client.

**Default Logon ID:** Type the default logon ID used to connect to your SQL Server database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

**Use NT Authentication:** Select this check box to enable Windows NT security. When enabled, the Default Logon ID field is inactive because Windows NT security passes the logon

ID and password. Selecting this check box also activates the corresponding check box on the Logon Dialog.

**Enable Quoted Identifiers:** Select this check box to enable quoted identifiers; that is, identifiers in SQL Server can be quoted using a quoting character. The default is not selected.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. A logon dialog box is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box"](#) on page 260 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 will display 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**.

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

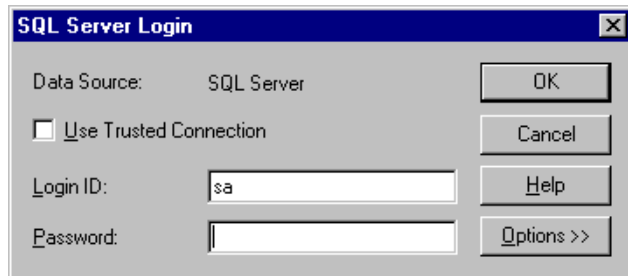
source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection string with alternate values.

---

## Connecting to a Data Source 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.

The SQL Server driver displays the SQL Server Login dialog box when you call an ODBC connection without specifying enough information for the driver to connect to an SQL Server.

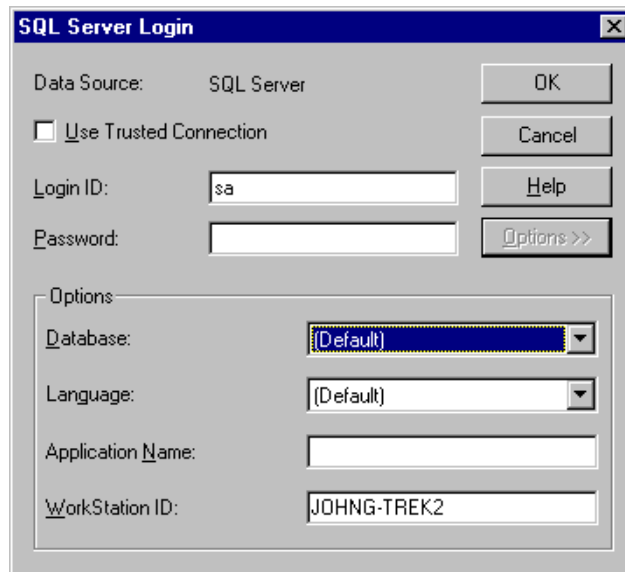


In this dialog box, perform the following steps:

- 1 Select the Use Trusted Connection check box to specify that the SQL Server driver request a secure (or trusted) connection to an SQL Server running on Windows NT. 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 SQL Server system administrator must have associated your Windows network ID with an SQL Server login ID.

Clear this box to specify that 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.

- 2 Type the 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.
- 3 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.
- 4 You can click **Options** to display the SQL Server Logon Options pane and specify the initial SQL Server database to which to connect.



**Database:** Type the default database to use on the connection. This overrides the default database specified for the login on the server. If no database is specified, the connection uses the default database specified for the login on the server.

**Language:** Type the national language to use for SQL Server system messages. The SQL Server must have the language installed. 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.

**Application Name:** Optionally, you can type the application name to be stored in the `program_name` column in the row for this connection in `master.dbo.sysprocesses`.

**Workstation ID:** Optionally, you can type the workstation ID to be stored in the `hostname` column in the row for this connection in `master.dbo.sysprocesses`.

- 5 Click **OK** to log on to the SQL Server database installed on the server you specified and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

The connection string has the form:

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

An example of a connection string for SQL Server is:

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

The following tables give the names and descriptions of the attributes.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

## Windows

---

**Table 12-1. SQL Server Connection String Attributes on Windows**

---

Attribute	Description
Address	The network address of the server running SQL Server. Used only if the Server keyword does not specify the network name of a server running SQL Server. Address is usually the network name of the server, but can be other names such as a pipe, or a TCP/IP port and socket address. For example, on TCP/IP: 199.199.199.5, 1433 or MYSVR, 1433.
AnsiNPW	AnsiNPW={yes   no}. Determines whether ANSI-defined behaviors are exposed.  When set to yes, the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation.  When set to no, ANSI-defined behaviors are not exposed.
APP	The name of the application calling SQLDriverConnect (optional). If specified, this value is stored in the master.dbo.sysprocesses column program_name and is returned by sp_who and the Transact-SQL APP_NAME function.

---

**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

---

Attribute	Description
AttachDBFileName	<p>The 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:</p> <p>AttachDBFileName=c:\\MyFolder\\MyDB.mdf</p> <p>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 SQLDriverConnect DATABASE parameter or the SQL_COPT_CURRENT_CATALOG connection attribute. If the database was previously attached, SQL Server will not reattach it; it will use the attached database as the default for the connection.</p>
AutoTranslate	<p>AutoTranslate={yes   no}. Determines how ANSI character strings are translated.</p> <p>When set to yes, 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.</p> <p>These conversions are performed on the client by the SQL Server ODBC driver. This requires that the same ANSI code page (ACP) used on the server be available on the client.</p>



**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

<b>Attribute</b>	<b>Description</b>
AutoTranslate (cont.)	<p>These settings have no effect on the conversions that occur for the following transfers:</p> <p>Unicode SQL_C_WCHAR client data sent to char, varchar, or text on the server.</p> <p>Char, varchar, or text server data sent to a Unicode SQL_C_WCHAR variable on the client.</p> <p>ANSI SQL_C_CHAR client data sent to Unicode nchar, nvarchar, or ntext on the server.</p> <p>Unicode char, varchar, or text server data sent to an ANSI SQL_C_CHAR variable on the client.</p> <p>When set to no, character translation is not performed.</p> <p>The SQL Server ODBC 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.</p> <p>If the client and SQL Server are using different ACPs, then extended characters can be misinterpreted.</p>
DATABASE	<p>The name of the default SQL Server database for the connection. If DATABASE is not specified, the default database defined for the login is used. The default database from the ODBC data source overrides the default database defined for the login. The database must be an existing database unless AttachDBFileName is also specified. If AttachDBFileName is specified, the primary file it points to is attached and given the database name specified by DATABASE.</p>

**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

Attribute	Description
Fallback <i>SQL Server 6.5 only</i>	<p data-bbox="486 314 1233 374">Fallback={yes   no}. Determines whether the driver attempts a connection to a fallback server.</p> <p data-bbox="486 392 1233 548">When set to yes, the driver attempts a connection to a fallback server if connection to a primary server fails. The login time-out (set with ODBC SQLSetConnectAttr, attribute SQL_ATTR_LOGIN_TIMEOUT) must be set for fallback to occur.</p> <p data-bbox="486 565 1233 652">When set to no, no attempt at a fallback connection is made. This option applies only to standby servers. It does not apply to a virtual server in a cluster/failover configuration.</p>
LANGUAGE	<p data-bbox="486 670 1233 826">The SQL Server language name (optional). SQL Server can store messages for multiple languages in sysmessages. If connecting to a SQL Server with multiple languages, This attribute specifies which set of messages are used for the connection.</p>
Network	<p data-bbox="486 843 1233 930">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.</p>
PWD	<p data-bbox="486 947 1233 1069">The password for the SQL Server login account specified in the UID parameter. PWD need not be specified if the login has a NULL password or when using Windows NT authentication (Trusted_Connection = yes).</p>
QueryLogFile	<p data-bbox="486 1086 1233 1138">The full path and file name of a file to be used for logging data about long-running queries.</p>
QueryLog_On	<p data-bbox="486 1173 1233 1225">QueryLog_On={yes   no}. Determines whether long-running query data is logged.</p> <p data-bbox="486 1242 1233 1303">When set to yes, logging long-running query data is enabled on the connection.</p>
QueryLogTime	<p data-bbox="486 1321 1233 1355">When set to no, long-running query data is not logged.</p> <p data-bbox="486 1364 1233 1487">A digit character string specifying the threshold (in milliseconds) for logging long-running queries. Any query that does not receive a response in the time specified is written to the long-running query log file.</p>

**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

Attribute	Description
QuotedID	<p>QuotedID={yes   no}. Determines whether QUOTED_IDENTIFIER is set ON or OFF for the connection.</p> <p>When set to yes, QUOTED_IDENTIFIER is set ON for the connection, and SQL Server uses the SQL-92 rules regarding the use of quotation marks in SQL statements.</p> <p>When set to no, QUOTED_IDENTIFIER is set OFF for the connection, and SQL Server uses the legacy Transact-SQL rules regarding the use of quotation marks in SQL statements.</p>
Regional	<p>Regional={yes   no}. Determines how currency, date, and time data are converted.</p> <p>When set to yes, the SQL Server 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.</p> <p>When set to no, the driver uses ODBC standard strings to represent currency, date, and time data that is converted to string data.</p>
SAVEFILE	<p>The name of an ODBC data source file into which the attributes of the current connection are saved if the connection is successful.</p>
SERVER	<p>The name of a server running SQL Server on the network. The value must be either the name of a server on the network, or the name of a SQL Server Client Network Utility advanced server entry. You can enter "(local)" as the server name on Windows NT to connect to a copy of SQL Server running on the same computer.</p>
StatsLogFile	<p>The full path and file name of a file used to record SQL Server driver performance statistics.</p>

---

**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

---

<b>Attribute</b>	<b>Description</b>
StatsLog_On	<p>StatsLog_On={yes   no}. Determines whether SQL Server driver performance data is available.</p> <p>When set to yes, SQL Server driver performance data is captured.</p> <p>When set to no, SQL Server driver performance data is not available on the connection.</p>
Trusted_Connection	<p>Trusted_Connection={yes   no}. Determines what information the SQL Server driver will use for login validation.</p> <p>When set to yes, the SQL Server driver uses Windows NT Authentication Mode for login validation. The UID and PWD keywords are optional.</p> <p>When set to no, the SQL Server driver uses a SQL Server username and password for login validation. The UID and PWD keywords must be specified.</p>
UID	<p>A valid SQL Server login account. UID need not be specified when using Windows NT authentication.</p>

**Table 12-1. SQL Server Connection String Attributes on Windows** (cont.)

Attribute	Description
UseProcForPrepare <i>SQL Server 6.5 and earlier only</i>	UseProcForPrepare={0   1   2}. Determines whether the SQL Server driver creates temporary stored procedures for SQLPrepare.  When set to 0, the SQL Server ODBC driver does not create temporary stored procedures for SQLPrepare.  When set to 1, the SQL Server ODBC driver creates temporary stored procedures when statements are prepared with SQLPrepare. The temporary stored procedures are not dropped until the connection is broken.  When set to 2, the SQL Server ODBC driver creates temporary stored procedures for SQLPrepare, but only one procedure is created per statement handle and the procedure is dropped when the statement handle becomes invalid or a new SQL statement is prepared.
WSID	The workstation ID. Typically, this is the network name of the computer on which the application resides (optional). If specified, this value is stored in the master.dbo.sysprocesses column hostname and is returned by sp_who and the Transact-SQL HOST_NAME function.

**UNIX****Table 12-2. SQL Server Connection String Attributes on UNIX**

Attribute	Description
Address	The network address of the server running SQL Server. This is required on UNIX and must be a TCP/IP port and socket address, for example, 199.199.199.5, 1433 or MYSVR, 1433.  NOTE: Address is not valid for the system information file.

**Table 12-2. SQL Server Connection String Attributes on UNIX** (cont.)

Attribute	Description
AnsiNPW	<p>AnsiNPW={yes   no}. Determines whether ANSI-defined behaviors are exposed.</p> <p>When set to yes, the driver uses ANSI-defined behaviors for handling NULL comparisons, character data padding, warnings, and NULL concatenation.</p> <p>When set to no, ANSI-defined behaviors are not exposed.</p>
APP	<p>The name of the application calling SQLDriverConnect (optional). If specified, this value is stored in the master.dbo.sysprocesses column program_name and is returned by sp_who and the Transact-SQL APP_NAME function.</p>
AppCodePage (ACP)	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>
DATABASE	<p>The name of the default SQL Server database for the connection. If DATABASE is not specified, the default database defined for the login is used. The default database from the ODBC data source overrides the default database defined for the login. The database must be an existing database unless AttachDBFileName is also specified. If AttachDBFileName is specified, the primary file it points to is attached and given the database name specified by DATABASE.</p>

**Table 12-2. SQL Server Connection String Attributes on UNIX** (cont.)

<b>Attribute</b>	<b>Description</b>
LANGUAGE	The SQL Server language name (optional). SQL Server can store messages for multiple languages in sysmessages. If connecting to a SQL Server with multiple languages, this attribute specifies which set of messages are used for the connection.
PWD	The password for the SQL Server login account specified in the UID parameter. PWD need not be specified if the login has a NULL password or when using Windows NT authentication (Trusted_Connection = yes).
QuotedID	<p>QuotedID={yes   no}. Determines whether QUOTED_IDENTIFIER is set ON or OFF for the connection.</p> <p>When set to yes, QUOTED_IDENTIFIER is set ON for the connection, and SQL Server uses the SQL-92 rules regarding the use of quotation marks in SQL statements.</p> <p>When set to no, QUOTED_IDENTIFIER is set OFF for the connection, and SQL Server uses the legacy Transact-SQL rules regarding the use of quotation marks in SQL statements.</p>
TDS	<p>TDS={7.0   4.2}. Determines the version of TDS.</p> <p>When set to 7.0, the driver connects to SQL Server 7.0 databases.</p> <p>When set to 4.2, the driver connects to SQL Server 6.5 databases.</p>
UID	<p>A valid SQL Server login account. UID need not be specified when using Windows NT authentication.</p> <p>NOTE: UID is not valid for the system information file; you must use LogonID.</p>

**Table 12-2. SQL Server Connection String Attributes on UNIX** (cont.)

Attribute	Description
UseProcForPrepare SQL Server 6.5 and earlier only	<p>UseProcForPrepare={0   1   2}. Determines whether the SQL Server driver creates temporary stored procedures for SQLPrepare.</p> <p>When set to 0, the SQL Server ODBC driver does not create temporary stored procedures for SQLPrepare.</p> <p>When set to 1, the SQL Server ODBC driver creates temporary stored procedures when statements are prepared with SQLPrepare. The temporary stored procedures are not dropped until the connection is broken.</p> <p>When set to 2, the SQL Server ODBC driver creates temporary stored procedures for SQLPrepare, but only one procedure is created per statement handle and the procedure is dropped when the statement handle becomes invalid or a new SQL statement is prepared.</p>
WSID	<p>The workstation ID. Typically, this is the network name of the computer on which the application resides (optional). If specified, this value is stored in the master.dbo.sysprocesses column hostname and is returned by sp_who and the Transact-SQL HOST_NAME function.</p>

---

## Unicode Support

The SQL Server driver supports unicode when connected to a version 7.0 SQL Server database.

The SQL Server driver maps the SQL Server data types as follows:

SQL Server Data Type	Mapped to. . .
Nchar	SQL_WCHAR
Ntext	SQL_WLONGVARCHAR
Sysname	SQL_WVARCHAR



# Data Types

[Table 12-3](#) shows how the SQL Server data types are mapped to the standard ODBC data types.

**Table 12-3. SQL Server Data Types**

SQL Server	ODBC
Binary	SQL_BINARY
Bit	SQL_BIT
Char	SQL_CHAR
Datetime	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
Nchar*	SQL_WCHAR
Ntext*	SQL_WLONGVARCHAR
Numeric	SQL_NUMERIC
Numeric() identity	SQL_NUMERIC
Nvarchar*	SQL_WVARCHAR
Real	SQL_REAL
Smalldatetime	SQL_TYPE_TIMESTAMP
Smallint	SQL_SMALLINT
Smallint identity	SQL_SMALLINT
Smallmoney	SQL_DECIMAL
Sysname	SQL_VARCHAR
Sysname*	SQL_WVARCHAR
Text	SQL_LONGVARCHAR

---

**Table 12-3. SQL Server Data Types (cont.)**


---

SQL Server	ODBC
Timestamp	SQL_VARBINARY
Tinyint	SQL_TINYINT
Tinyint identity	SQL_TINYINT
Uniqueidentifier*	SQL_GUID
Varbinary	SQL_VARBINARY
Varchar	SQL_VARCHAR

---

\* Supported by SQL Server 7 only.

---

## Isolation and Lock Levels Supported

SQL Server supports isolation levels 0 (read uncommitted), 1 (read committed), 2 (repeatable read), and 3 (serializable). SQL Server supports row-level and table-level locking. See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.

---

## ODBC Conformance Level

The SQL Server driver supports ODBC conformance level 2.

---

## Number of Connections and Statements Supported

The SQL Server database system supports multiple connections. With two-phased commit, SQL Server supports multiple statements per connection. Otherwise, SQL Server supports a single statement per connection if `SQL_AUTOCOMMIT` is 0 and multiple statements per connection if `SQL_AUTOCOMMIT` is 1.



# 13 Connect ODBC for SQL Server 6

Connect ODBC for SQL Server 6 (the "SQL Server 6 driver") supports the SQL Server 6.5 database system available from Microsoft in the Windows environments. It also supports the SQL Server 7.0 and SQL Server 2000 database systems, but with 6.5 functionality only.

See "[Environment-Specific Information](#)" on page 33 for detailed information about the Windows environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the SQL Server 6 driver.

---

## Driver Requirements

To use the SQL Server 6 driver, you must have the appropriate Microsoft SQL Server DB-Library and Net-Library version installed (version 6.5 for access to 6.5 DBMS; version 7.0 for access to 7.0 DBMS).

The SQL Server 6 driver requires Microsoft client software; it does *not* work with Sybase System 10 or 11 software.

Your database must support catalog stored procedures.

The DB-Library is NTWDBLIB.DLL. The Net-Library you need depends on the network protocol used to connect to the SQL Server. For example, Named Pipes requires DBNMPNTW.DLL, and TCP/IP requires DBMSSOCN.DLL. Contact your Microsoft SQL Server vendor to obtain the appropriate DB-Library and Net-Library.

You must have NTWDBLIB.DLL on your path or in your Windows 9x or Windows Me \SYSTEM directory, or Windows NT or Windows 2000 \SYSTEM32 directory.

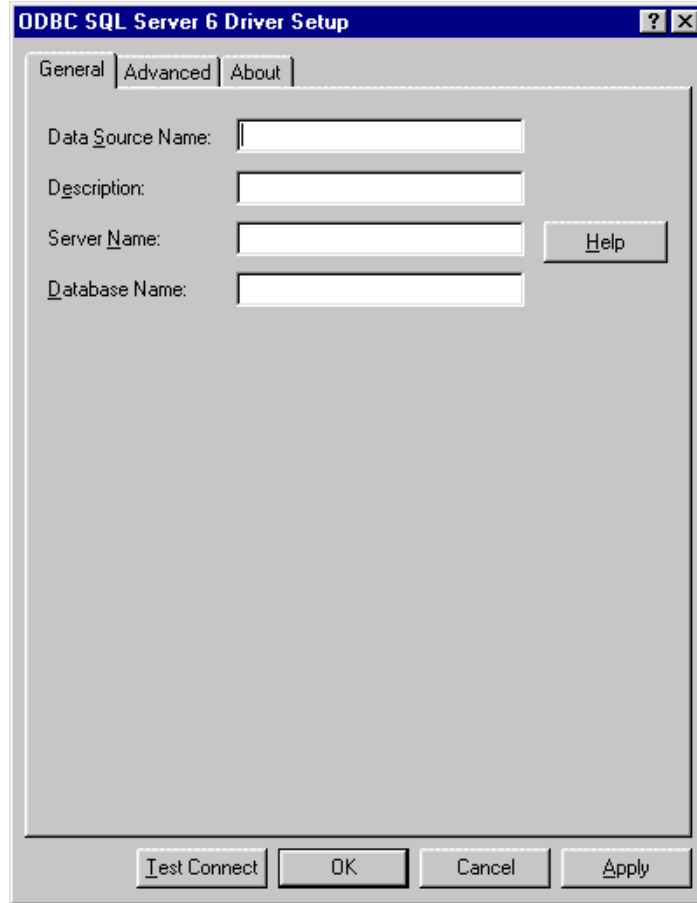
---

## Configuring Data Sources

Data sources are configured and modified through the ODBC Administrator. To configure a SQL Server 6 data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring a new data source, click **Add**. A list of installed drivers appears. Select SQL Server 6 and click **Finish** to display the ODBC SQL Server 6 Driver Setup dialog box.

If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC SQL Server 6 Driver Setup dialog box.



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

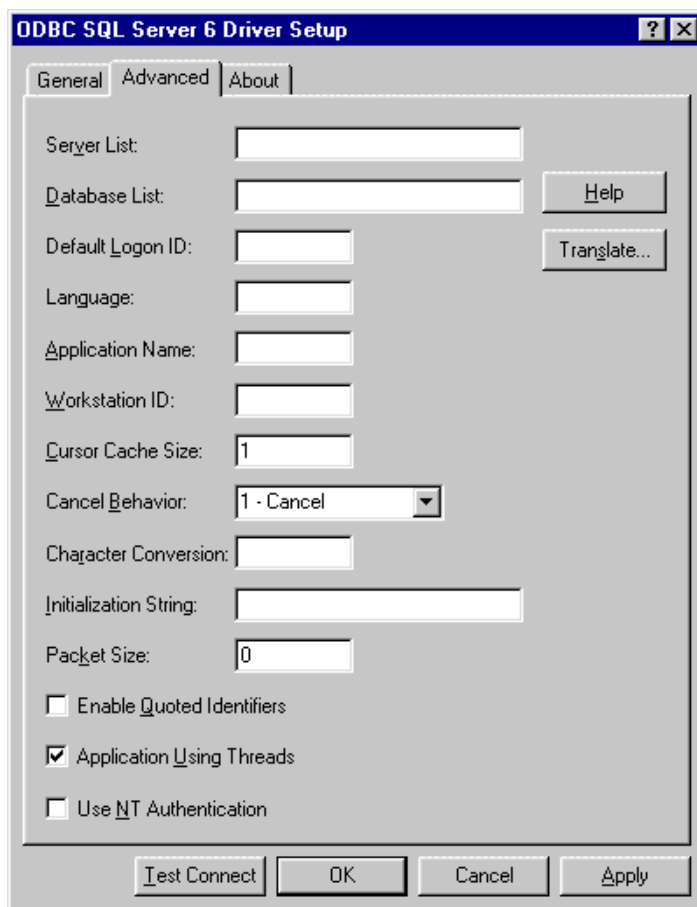
**Data Source Name:** Type a string that identifies this SQL Server data source configuration in the system information. Examples include "Accounting" or "SQL Server-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "SQL Server on Server number 1."

**Server Name:** Type the name of the server that contains the database you want.

**Database Name:** Type the name of the database to which you want to connect by default. If you do not specify a value, the default database defined by SQL Server is used.

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





On this tab, provide any of the following optional information; then, click **Apply**.

**Server List:** Type a comma-separated list of servers that will appear in the Logon dialog box.

**Database List:** Type the databases that will be available in the SQL Server Logon Options dialog box. Separate the names with commas.

**Default Logon ID:** Type the default logon ID used to connect to your SQL Server database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

**Language:** Type the national language to be used by the client. The default is English.

**Application Name:** Type the name SQL Server uses to identify your application.

**Workstation ID:** Type the workstation ID used by the client.

**Cursor Cache Size:** Type the number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources. The default is 1 (one cursor).

**Cancel Behavior:** Select an integer value of 0, 1, or 2 that specifies how a previously executed statement should be canceled.

When set to 0, all of the remaining records are fetched if the statement was a Select.

When set to 1, the statement is canceled by calling `dbcancel`. This is the default and should be used if `dbcancel` is supported in your client/server configuration.

When set to 2, the connection is closed to the server for the statement. Use this value only if `dbcancel` is not supported for your configuration and the performance of fetching all remaining records is unacceptable.

**Character Conversion:** Type a value that controls the character set conversion between SQL Server and a client application. If you omit this value, no character conversion takes place between the client and server.

Common values include `iso_1` for ISO-8859-1, `cp850` for Code Page 850, `roman8` for Roman8 character set, and `SJIS` for a Japanese character set. See your SQL Server documentation for a complete list of values.

**Initialization String:** Type a string containing one or more SQL Server commands that you want to run when the data source connection is initialized. Multiple commands must be separated by a semicolon (;).

**Packet Size:** Type a value of -1, 0, or  $x$  that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.

When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the system information.

When set to 0 (the default), the driver uses the default packet size as specified in the server configuration.

When set to  $x$ , an integer from 1 to 10, which indicates a multiple of 512 bytes (for example, 6 means to set the packet size to  $6 * 512 = 3072$  bytes).

**NOTE:** The ODBC specification identifies a connect option, `SQL_PACKET_SIZE`, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, the ODBC connect option will take precedence.

**Enable Quoted Identifiers:** Select this check box to enable quoted identifiers; that is, identifiers in SQL Server can be quoted using a quoting character. By default, the check box is not selected.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Use NT Authentication:** Select this check box to enable Windows NT security. When enabled, the Default Logon ID field is inactive because Windows NT security passes the logon ID and password. Selecting this check box also activates the corresponding check box on the Logon Dialog.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. A logon dialog box is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box" on page 284](#) 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 will display 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].

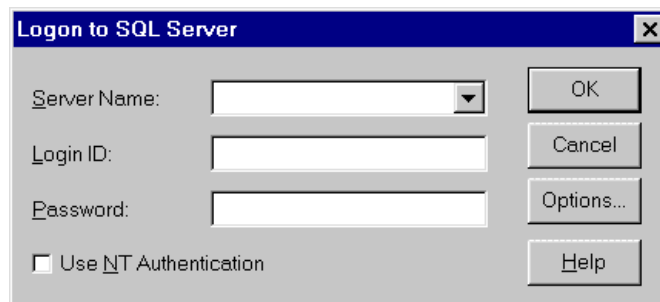
Click **OK**.

- 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.

---

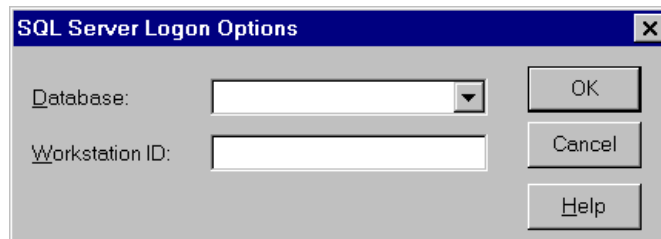
## Connecting to a Data Source Using a Logon Dialog Box

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. For SQL Server, the dialog box is as follows:



In this dialog box, perform the following steps:

- 1 Type the name of the server containing the SQL Server database tables you want to access (case-sensitive) or select the name from the Server Name drop-down list, which displays the server names you specified in the ODBC SQL Server Driver Setup dialog box.
- 2 If required, type your case-sensitive login ID.
- 3 If required, type your case-sensitive password for the system.
- 4 Use NT Authentication enables Windows NT security. When enabled, the Login ID and Password fields are inactive because Windows NT security passes the login ID and password. Activating this check box also activates the corresponding check box on the Advanced Tab.
- 5 You can click **Options** to display the SQL Server Logon Options dialog box and specify the initial SQL Server database to connect to and the name of your workstation.



- 6 Click **OK** to log on to the SQL Server database installed on the server you specified and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for SQL Server 6 is:

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

[Table 13-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 13-1. SQL Server 6 Connection String Attributes**

Attribute	Description
ApplicationName (APP) ApplicationUsing Threads (AUT)	<p>The name SQL Server uses to identify your application.</p> <p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe. When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
Cancel (CAN)	<p>Cancel={0   1   2}. Determines how a previously executed statement should be canceled.</p> <p>When set to 0, all remaining records are fetched if the statement was a Select.</p> <p>When set to 1 (the initial default), the statement is canceled by calling dbcanceled. Set to 1 if dbcanceled is supported in your client/server configuration.</p> <p>When set to 2, the connection is closed to the server for the statement. Set to 2 only if dbcanceled is not supported for your configuration and the performance of fetching all remaining records is unacceptable.</p>
CharConv (CC)	<p>A value that controls the character set conversion between SQL Server and a client application. Common values include:</p> <ul style="list-style-type: none"> <li>■ iso_1 for ISO-8859-1</li> <li>■ cp850 for Code Page 850</li> <li>■ roman8 for the Roman8 character set</li> <li>■ SJIS for a Japanese character set.</li> </ul> <p>See your SQL Server documentation for a complete list of values.</p>
CursorCacheSize (CCS)	<p>The number of cursors the cursor cache can hold. The driver creates a cache of statements; each statement represents an open connection to SQL Server. The cursor cache increases performance but uses database resources.</p> <p>The initial default is 1.</p>
Database (DB)	<p>The name of the database to which you want to connect.</p>

---

**Table 13-1. SQL Server 6 Connection String Attributes** (cont.)

---

<b>Attribute</b>	<b>Description</b>
DataSourceName (DSN)	A string that identifies a SQL Server data source configuration in the system information. Examples include "Accounting" or "SQL Server-Serv1."
EnableQuoted Identifiers (EQI)	EnableQuotedIdentifiers={0   1}. Determines whether identifiers in SQL Server can be quoted using a quoting character.  When set to 0 (the initial default), quoted identifiers are disabled.  When set to 1, quoted identifiers are enabled.
InitializationString (IS)	A string that contains one or more SQL Server commands that you want to run when the data source connection is initialized. Multiple commands must be separated by a semicolon (;).
Language (LANG)	The national language to be used by the client.  The initial default is English.
LogonID (UID)	The case-sensitive logon ID used to connect to your SQL Server database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.



**Table 13-1. SQL Server 6 Connection String Attributes** (cont.)

Attribute	Description
PacketSize (PS)	<p>PacketSize={-1   0   <math>x</math>}. Determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.</p> <p>When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the system information.</p> <p>When set to 0 (the initial default), the driver uses the default packet size as specified in the server configuration.</p> <p>When set to <math>x</math>, an integer from 1 to 10, the driver uses a packet size represented by <math>x</math> times 512 bytes. For example, PacketSize=6 means to set the packet size to <math>6 * 512</math> bytes (3072 bytes).</p> <p>NOTE: The ODBC specification specifies a connect option, SQL_PACKET_SIZE, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, the ODBC connect option will take precedence.</p>
Password (PWD)	A case-sensitive password.
ServerName (SRVR)	The name of the server containing the SQL Server tables you want to access.
UseNTAuthentication (UNA)	<p>UseNTAuthentication={0   1}. Enables Windows NT security.</p> <p>When set to 0 (the initial default), Windows NT security is disabled.</p> <p>When set to 1, Windows NT security is enabled, and the LogonID and Password attributes are disabled.</p>
WorkstationID (WKID)	The workstation ID used by the client.

## Data Types

Table 13-2 shows how the SQL Server 6 data types are mapped to the standard ODBC data types.

**Table 13-2. SQL Server 6 Data Types**

SQL Server 6	ODBC
Binary	SQL_BINARY
Bit	SQL_BIT
Char	SQL_CHAR
Datetime	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
Sysname	SQL_VARCHAR
Text	SQL_LONGVARCHAR
Timestamp	SQL_VARBINARY
Tinyint	SQL_TINYINT
Tinyint identity	SQL_TINYINT

---

**Table 13-2. SQL Server 6 Data Types (cont.)**


---

SQL Server 6	ODBC
Varbinary	SQL_VARBINARY
Varchar	SQL_VARCHAR

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on STATIC cursors. For example:

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

- 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 regarding Step 1:** 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.
```

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, SQL\_PERSIST\_AS\_XML. A client application must call SQLSetStmtAttr with this new attribute as

an argument. See the following table for the definition of valid arguments for `SQLSetStmtAttr`.

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

---

## Isolation and Lock Levels Supported

SQL Server supports isolation levels 1 (read committed) and 3 (serializable). SQL Server supports page-level locking. See [Appendix D, “Locking and Isolation Levels” on page 439](#) for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the SQL Server 6 driver.

In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges
- SQLSetPos

Scrollable cursors are supported with `SQLExtendedFetch` and `SQLFetchScroll`. The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The SQL Server database system supports multiple connections. With two-phased commit, SQL Server supports multiple statements per connection. Otherwise, SQL Server supports a single statement per connection if `SQL_AUTOCOMMIT` is 0 and multiple statements per connection if `SQL_AUTOCOMMIT` is 1.



# 14 Connect ODBC for SQLBase

Connect ODBC for SQLBase (the "SQLBase driver") supports the Centura Software SQLBase database system in the Windows environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the SQLBase driver.

---

## Driver Requirements

To communicate with the SQLBase server, the SQLBase driver requires the SQLBase 7.0 client software with PTF4 or higher, which includes the following shared libraries: SQLWNTM.DLL, SQLNGCI.DLL, and a communication DLL (for example, SQLNPIPE.DLL for Named Pipes). The directory containing these files must be on your path.

You must have version 6.x or higher of SQLBase.

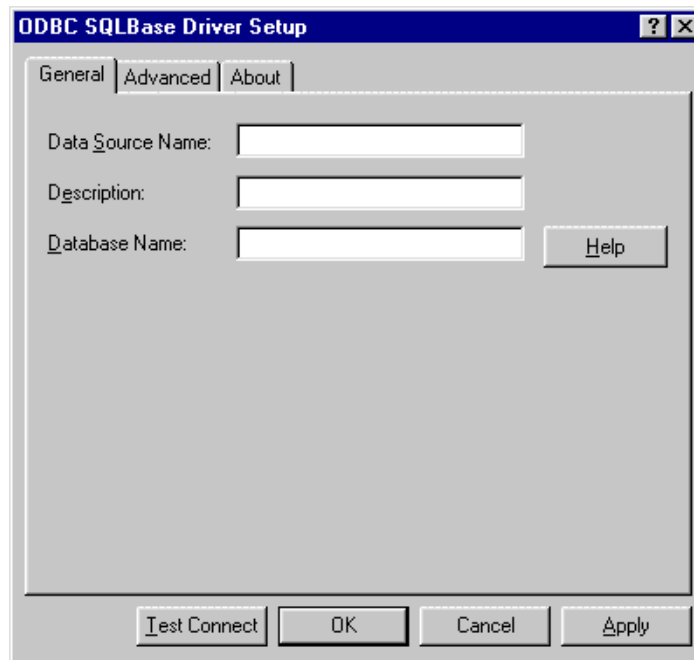
---

## Configuring Data Sources

Data sources are configured and modified through the ODBC Administrator. To configure a SQLBase data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC SQLBase Driver Setup dialog box.

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



**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.



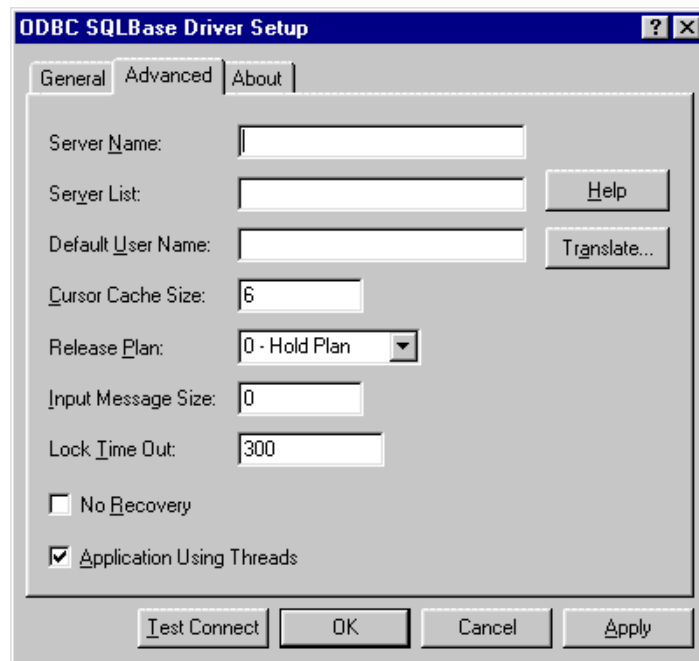
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this SQLBase data source configuration in the system information. Examples include "Accounting" or "SQLBase-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "SQLBase on Server number 1."

**Database Name:** Type the name of the database to which you want to connect by default.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Server Name:** Type the name of the server that contains the desired database. Type the word Local if you are using the local server.

The server name is not required to log on. The dialog box appears more quickly if you do not specify a server name. If you do specify a server name, the Database Name drop-down list in the Logon dialog box is populated with the names of the databases available on that server.

**Server List:** Type a comma-separated list of servers that will appear in the Logon dialog box. Type Local to add the local server to the list.

**Default User Name:** Type the default user name used to connect to your SQLBase database. A user name is required only if security is enabled on your database. Your ODBC application may override this value or you may override this value in the Logon dialog box or connection string.

**Cursor Cache Size:** Type the number of cursors the cursor cache can hold. The default is 6.

**Release Plan:** Select a value of 0 or 1 that determines whether a lock is maintained on a table when the cursors accessing the table are freed.

When set to 0, Hold Plan, no locks on tables are freed. This value is the default.

When set to 1, Free Plan, locks are freed.

Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table.

**Input Message Size:** Type the number of bytes in the input message buffer. The default is determined by SQLBase. Increasing this value retrieves more records across the network in a single fetch.

**Lock Time Out:** Type the number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 (wait forever) to 1800; the default is 300.

**No Recovery:** Select this check box to disable transaction recovery. Selecting this box is dangerous because your database can become inconsistent in the event of a system failure.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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. A logon dialog box is displayed; see ["Connecting to a Data Source Using a Logon Dialog Box"](#) on page 300 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 will display 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].
```

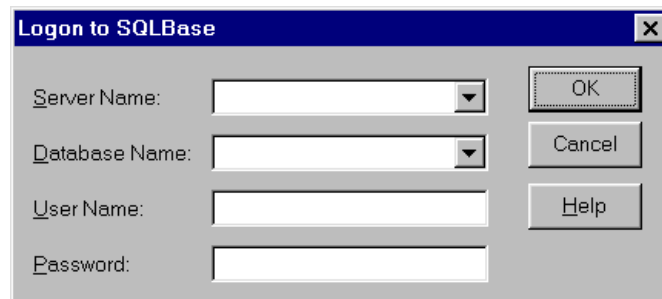
Click **OK**.

- 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.

---

## Connecting to a Data Source Using a Logon Dialog Box

Some ODBC applications display a Logon dialog box when you are connecting to a data source. For SQLBase, the dialog box is as follows:



In this dialog box, perform the following steps:

- 1 Optionally, type the name of the server containing the SQLBase database tables you want to access or select the name from the Server Name drop-down list, which displays

the server names you specified in the Setup dialog box. Specify Local to access a local SQLBase database.

- 2 Type the name of the database you want to access. If you specified a server name, you can select the name from the Database Name drop-down list.
- 3 If required, type your user name.
- 4 If required, type your password.
- 5 Click **OK** to complete the logon and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for SQLBase is:

```
DSN=SQLBASE TBL;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```

[Table 14-1](#) gives the long and short names for each attribute, as well as a description.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---

**Table 14-1. SQLBase Connection String Attributes**

---

<b>Attribute</b>	<b>Description</b>
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
CursorCacheSize (CCS)	<p>The number of cursors the cursor cache can hold. The cursor cache increases performance and uses few database resources.</p> <p>The initial default is 6.</p>
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies a SQLBase data source configuration in the system information. Examples include "Accounting" or "SQLBase-Serv1."
InputMessageSize (IMS)	<p>An integer value that determines the size in bytes of the input message buffer. Increasing this value retrieves more records across the network in a single fetch.</p> <p>The initial default is determined by SQLBase.</p>
LockTimeOut (LTO)	<p>The number of seconds SQLBase should wait for a lock to be freed before raising an error. Values can be -1 to 1800; -1 means wait forever.</p> <p>The initial default is 300 seconds.</p>
LogonID (UID)	The default logon ID (user name) used to connect to your SQLBase database. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.

**Table 14-1. SQLBase Connection String Attributes** (cont.)

Attribute	Description
NoRecovery (NR)	NoRecovery={0   1}. Enables transaction recovery. When set to 0 (the initial default), recovery is enabled. When set to 1, recovery is disabled. Performance is improved, but this setting is dangerous because your database can become inconsistent in the event of a system crash. See your SQLBase documentation for information on this option.
Password (PWD)	A case-sensitive password.
ReleasePlan (RP)	ReleasePlan={0   1}. Determines whether a lock is maintained on a table when the cursors accessing the table are freed. Freeing the lock on the table results in a request to the server, which can decrease performance but will always allow you to Drop or Alter the table. When set to 0 (the initial default), no locks on tables are freed. When set to 1, locks are freed.
ServerName (SRVR)	The name of the server containing the SQLBase database tables you want to access. Specify ServerName=Local if you are using the local server.
Servers (SRVRLIST)	A comma-separated list of servers with which to prompt the user in a Logon dialog box. Specify Local to add the local server to the list.

---

## Data Types

[Table 14-2](#) shows how the SQLBase data types are mapped to the standard ODBC data types.

**Table 14-2. SQLBase Data Types**

SQLBase	ODBC
Char	SQL_VARCHAR
Date	SQL_TYPE_DATE

---

**Table 14-2. SQLBase Data Types**

---

<b>SQLBase</b>	<b>ODBC</b>
Decimal	SQL_DECIMAL
Double Precision	SQL_DOUBLE
Integer	SQL_INTEGER
Long Varchar	SQL_LONGVARCHAR
Number	SQL_DOUBLE
Real	SQL_REAL
Smallint	SQL_SMALLINT
Time	SQL_TYPE_TIME
Timestamp	SQL_TYPE_TIMESTAMP
Varchar	SQL_VARCHAR

---

## Isolation and Lock Levels Supported

SQLBase supports isolation levels 1 (read committed, the default) and 3 (serializable). SQLBase also supports an alternative isolation level 1 called cursor stability. Your ODBC application can use this isolation level by calling `SQLSetConnectOption(1040,1)`.

SQLBase supports page-level locking.

See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.



---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the SQLbase driver. In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedures
- SQLSetPos
- SQLTablePrivileges

The driver also supports backward and random fetching in `SQLExtendedFetch` and `SQLFetchScroll`. It supports the core SQL grammar.

---

## Number of Connections and Statements Supported

The SQLBase database system supports multiple connections and multiple statements per connection.



# 15 Connect ODBC for Sybase ASE

Connect ODBC for Sybase ASE (the "Sybase ASE driver") supports Adaptive Server 11.0 and higher database systems from Sybase in the Windows and UNIX (except HP-UX 10.20) environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the Sybase ASE driver.

---

## Driver Requirements

There are no client requirements for the Sybase ASE driver.

---

## Configuring Data Sources

On Windows, data sources are configured and modified through the ODBC Administrator.

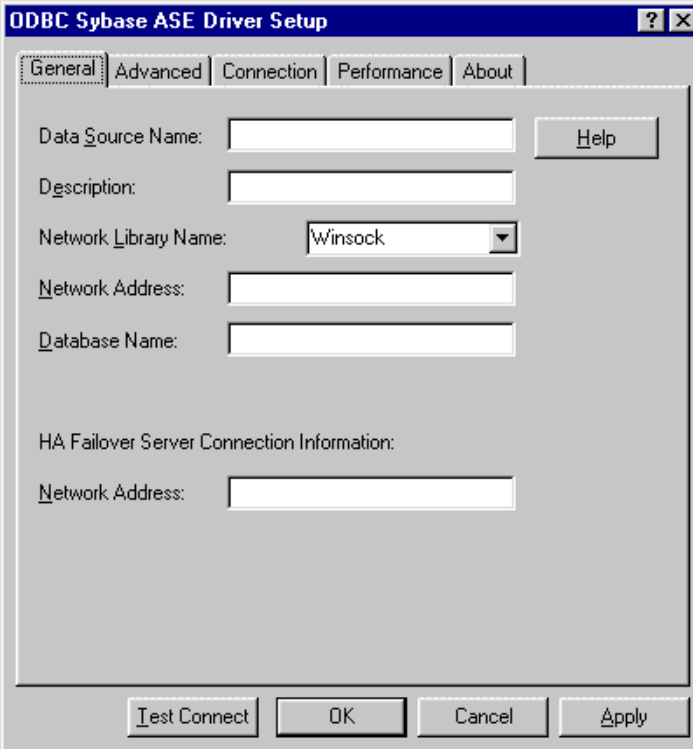


NOTE: In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 15-1 on page 320](#). You must also edit this file to perform a translation. See [Appendix H, "The UNIX Environments" on page 473](#) for information about editing the file.

**To configure a Sybase ASE data source:**

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Sybase ASE Driver Setup dialog box.

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



The image shows a screenshot of the "ODBC Sybase ASE Driver Setup" dialog box. The dialog has a title bar with a question mark and a close button. Below the title bar are five tabs: "General", "Advanced", "Connection", "Performance", and "About". The "General" tab is selected and active. It contains several input fields and a dropdown menu. The fields are: "Data Source Name:" (text box), "Description:" (text box), "Network Library Name:" (dropdown menu showing "Winsock"), "Network Address:" (text box), "Database Name:" (text box), and "HA Failover Server Connection Information:" (text box) with a "Network Address:" sub-field. A "Help" button is located to the right of the "Data Source Name" field. At the bottom of the dialog are four buttons: "Test Connect", "OK", "Cancel", and "Apply".

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

**Data Source Name:** Type a string that identifies this Sybase ASE data source configuration in the system information. Examples include "Accounting" or "Sys10-Serv1."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Database" or "System 10 on Server number 1."



**Network Library Name:** Select the name of the network library. This specifies which network protocol to use. The values are Winsock and NamedPipes. The default is Winsock. This option has no effect on UNIX; on UNIX, TCP/IP is used.

**Network Address:** Type the network address. The value you specify depends on which network protocol is chosen under Network Library Name and on the Sybase server. If you have chosen Winsock for the Network Library Name, specify an IP address as follows: "<servername or IP address>, <port number>". For example, if your network supports named servers, you may specify an address such as "Sybaseserver, 5000". You may also specify the IP address directly such as "199.226.224.34, 5000".

If you have chosen NamedPipes as the network protocol, you must specify the pipe address of the server. For example, "\\machine1\sybase\pipe\query".

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

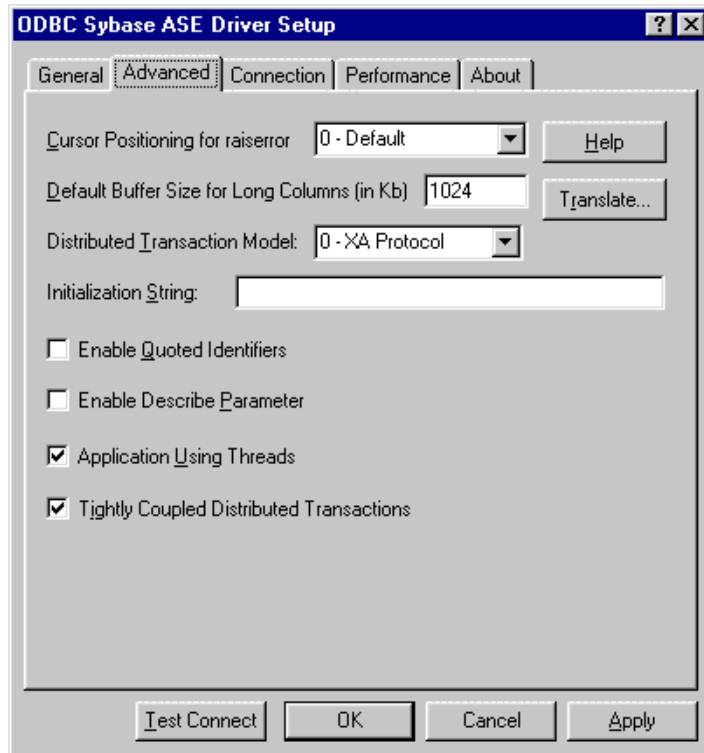
#### **HA Failover Server Connection Information/Network**

**Address:** Type 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 HA Failover server specified by this attribute. This option is valid only for Sybase ASE

version 12 servers that have the High Availability Failover feature enabled.

See the previous description of the Network Address option for an explanation of valid values.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Cursor Positioning for raiserror:** Select a value of 0 or 1 that specifies when the error is returned and where the cursor is positioned when raiserror is encountered.

When set to 0 (the default), raiserror is handled separately from surrounding statements. The error is returned when

raiserror is processed via SQLExecute, SQLExecDirect, or SQLMoreResults. The result set is empty.

When set to 1 (MS compatible), raiserror is handled with the next statement. The error is returned when the next statement is processed; the cursor is positioned on the first row of subsequent result set. This could result in multiple raiserrors being returned on a single execute.

**Default Buffer Size for Long Columns (in Kb):** Type an integer value that specifies the maximum length of data fetched from a TEXT or IMAGE column. The value must be in multiples of 1024 (for example, 1024, 2048. The default is 1024 KB. You will need to increase this value if the total size of any long data exceeds 1 MB.



**Distributed Transaction Model:** Select a model to use for distributed transaction support—either XA Protocol or Native OLE.

**Initialization String:** Enter a semicolon-separated list of Sybase language commands that will be issued immediately after connection.

**Enable Quoted Identifiers:** Select this check box to allow support of quoted identifiers.

**Enable Describe Parameter:** Select this check box to enable the SQLDescribeParam function, which allows an application to describe parameters in SQL statements and in stored procedure calls. To use this option, Prepare Method must be set to 0 or 1, and the SQL statement must not include long parameters. This option should be selected when using Microsoft Remote Data Objects (RDO) to access data.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. You can clear this check box when using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.



**Tightly Coupled Distributed Transactions:** Select this check box to use tightly coupled distributed transactions when connected to a Sybase ASE version 12 database and to ensure that multiple connections within the same distributed transaction do not obey each other's locks.

When this check box is not selected, the overall performance of the driver is better, but multiple connections within the same distributed transaction may hang each other because the connections do not obey each other's locks.

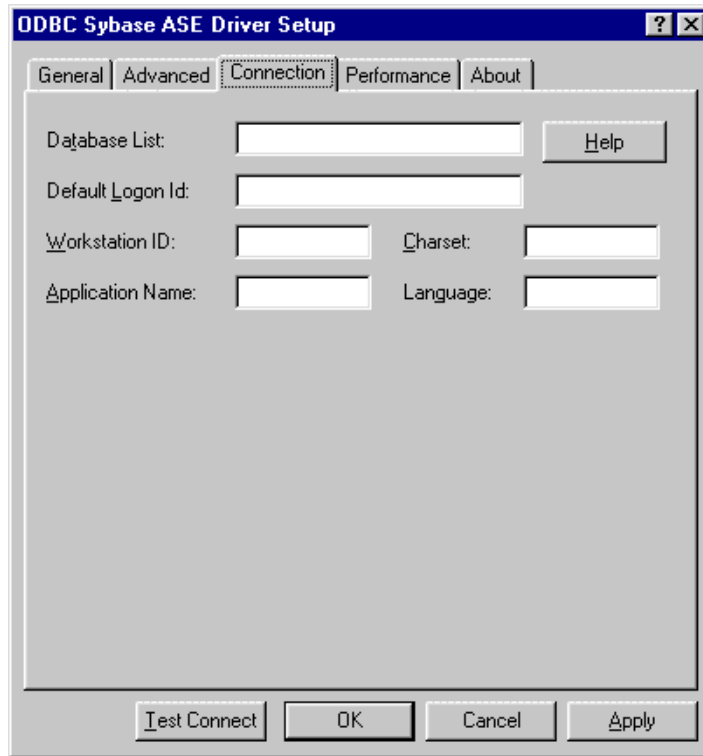
This option is valid only when the driver is enlisted in a distributed transaction and when it is connected to a Sybase ASE version 12 database. Otherwise, this option is ignored.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.



- 5 Optionally, click the **Connection** tab to specify data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

**Database List:** Type the databases that appear in the logon dialog box. Separate the names with commas.

**Default Logon ID:** Type the default logon ID used to connect to your Sybase database. This ID is case-sensitive. A logon ID is required only if security is enabled for the database you are connecting to. Your ODBC application may override this value or you can override this value in the logon dialog box or connection string.

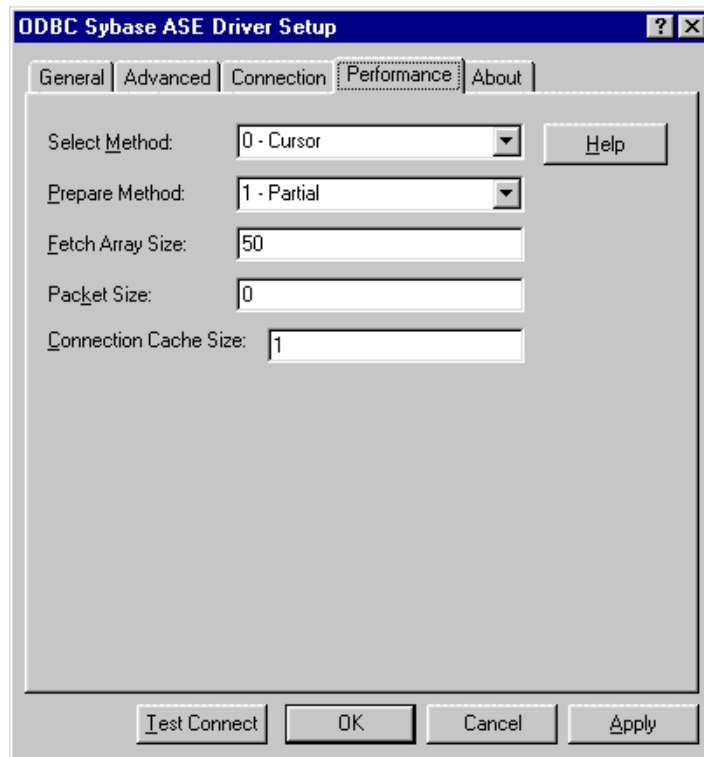
**Workstation ID:** Type the workstation ID used by the client.

**Application Name:** Type the name used by Sybase to identify your application.

**Charset:** Type the name of a character set. This character set must be installed on the Sybase server. The default is the setting on the Sybase server. For this driver to support Unicode, this attribute must be set to UTF-8. Refer to the Sybase server documentation for a list of valid character set names.

**Language:** Type the national language. This language must be installed on the Sybase server. The default is English.

- Optionally, click the **Performance** tab to specify performance data source settings.



On this tab, provide any of the following optional information; then, click **Apply**.

**Select Method:** Select a value of 0 or 1 that determines whether database cursors are used for Select statements. When set to 0, the default, database cursors are used; when set to 1, Select statements are run directly without using database cursors. A setting of 1 limits the data source to one active statement.

**Prepare Method:** Select a value of 0, 1, 2, or 3 that determines whether stored procedures are created on the server for calls to SQLPrepare.

When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in decreased performance when processing statements that do not contain parameters.

When set to 1 (the initial default), the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and run directly at the time of SQLExecute.

When set to 2, 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.

When set to 3, 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.

**Fetch Array Size:** Type the number of rows the driver retrieves when fetching from the server. This is not the number of rows given to the user. The default is 50 rows.

**Packet Size:** Type a value of -1, 0, or x that determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.

When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the system information.

When set to 0, the default, the driver uses the default packet size as specified in the Sybase server configuration.

When set to  $x$ , an integer from 1 to 1024, the driver uses a packet size represented by  $x$  times 512 bytes. For example, "6" means to set the packet size to  $6 * 512$  bytes (3072 bytes).

To take advantage of this connection attribute, you must configure the Sybase server for a maximum network packet size greater than or equal to the value you specified for PacketSize. For example:

```
sp_configure "maximum network packet size", 5120
reconfigure
Restart Sybase Server
```

**NOTE:** The ODBC specification identifies a connect option, `SQL_PACKET_SIZE`, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, they have been defined as mutually exclusive. If `PacketSize` is specified, you will receive a message "Driver Not Capable" if you attempt to call `SQL_PACKET_SIZE`. If you do not set `PacketSize`, then application calls to `SQL_PACKET_SIZE` are accepted by the driver.

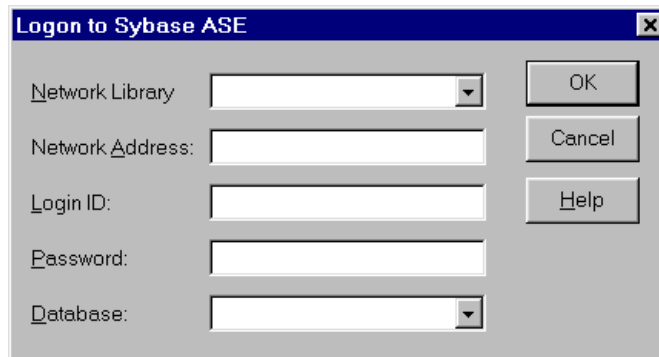
**Connection Cache Size:** Type a value that determines the number of connections that the connection cache can hold. The default Connection Cache setting is 1. To set the connection cache, you must set the Select Method option to 1 - Direct. Increasing the connection cache may increase performance of some applications but requires additional database resources.

- 7 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 is displayed; see [“Connecting to a Data Source Using a Logon Dialog Box” on page 318](#) 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 will display an appropriate error message. Click **OK**.
- 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.

---

## Connecting to a Data Source 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. For Sybase ASE, the dialog box is as follows:



In this dialog box, perform the following steps:

- 1 In the Network Library field, select the name of the network library. This specifies which network protocol to use. Valid values are Winsock and NamedPipes.
- 2 In the Network Address field, type the network address, which depends on which network protocol is chosen under Network Library and on the Sybase server. If you have chosen Winsock, specify an IP address as follows: "<servername or IP address>, <port number>". For example, if your network supports named servers, you may specify an address such as "Sybaseserver, 5000". You may also specify the IP address directly such as "199.226.224.34, 5000".

If you have chosen NamedPipes as the network protocol, you must specify the pipe address of the server. For example, "\\machine1\sybase\pipe\query".

- 3 If required, type your case-sensitive login ID.
- 4 If required, type your case-sensitive password for the system.
- 5 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 you specified in the ODBC Sybase ASE Driver Setup dialog box.
- 6 Click **OK** to complete the logon and to update the values in the system information.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for Sybase ASE is:

```
DSN=SYS10 TABLES;SRVR=QESRVR;DB=PAYROLL;UID=JOHN;PWD=XYZZY
```




[Table 15-1](#) gives the long and short names for each attribute, as well as a description. To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, “The UNIX Environments” on page 473](#) for information about this file.

The defaults listed in the table are initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

---


**Table 15-1. Sybase ASE Connection String Attributes**

---

Attribute	Description
AppCodePage (ACP)	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>
 ApplicationName (APP) ApplicationUsing Threads (AUT)	<p>The name used by Sybase to identify your application.</p> <p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you can set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>
ArraySize (AS)	<p>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 increases performance by reducing network traffic.</p> <p>The initial default is 50 rows.</p>
Charset (CS)	<p>The name of a character set. This character set must be installed on the Sybase server. The default is the setting on the Sybase server. For this driver to support Unicode, this attribute must be set to UTF-8. Refer to the Sybase server documentation for a list of valid character set names.</p>



**Table 15-1. Sybase ASE Connection String Attributes** (cont.)

<b>Attribute</b>	<b>Description</b>
CursorCacheSize (CCS)	<p>The number of connections that the connection cache can hold. To set the connection cache, you must set the SelectMethod attribute to 1. Increasing the connection cache may increase performance of some applications but requires additional database resources.</p> <p>The initial default is 1 (one cursor).</p>
Database (DB)	The name of the database to which you want to connect.
DataSourceName (DSN)	A string that identifies a single connection to a Sybase database. Examples include "Accounting" or "Sys10-Serv1."
DefaultLongData BuffLen (DLDBL)	<p>An integer value that specifies, in 1024-byte multiples, the maximum length of data fetched from a TEXT or IMAGE column. You will need to increase this value if the total size of any long data exceeds 1 MB.</p> <p>The default is 1024.</p>
DistributedTransaction Model (DTM)	<p>DistributedTransactionModel={XA Protocol   Native OLE}. Determines which model is used for distributed transaction support. The initial default is XA Protocol.</p>
	
EnableDescribe Param (EDP)	<p>EnableDescribeParam={0   1}. Determines whether the ODBC API function SQLDescribeParam is enabled.</p> <p>When set to 0 (the initial default), SQLDescribeParam is disabled.</p> <p>When set to 1, SQLDescribeParam is enabled, which allows an application to describe parameters in SQL statements and in stored procedure calls. To use this option, OptimizePrepare must be set to 0 or 1, and the SQL statement must not include long parameters. This attribute should be set to 1 when using Microsoft Remote Data Objects (RDO) to access data.</p>
EnableQuoted Identifiers (EQI)	<p>EnableQuotedIdentifiers={0   1}. Enables quoted identifiers.</p> <p>When set to 0 (the initial default), quoted identifiers are disabled.</p> <p>When set to 1, quoted identifiers are enabled.</p>

**Table 15-1. Sybase ASE Connection String Attributes** (cont.)

Attribute	Description
FailoverNetworkAddress (FNA)	<p>Specifies the 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 HA Failover server specified by this attribute. This attribute is valid only for Sybase ASE version 12 servers that have the High Availability Failover feature enabled.</p> <p>See the previous description of the Network Address attribute for an explanation of valid values.</p>
InitializationString (IS)	<p>InitializationString={<i>Sybase set commands</i>;...}. Supports the execution of Sybase commands at connect time. Multiple commands must be separated by semicolons.</p>
Language (LANG)	<p>The national language. This language must be installed on the Sybase server.</p> <p>The initial default is English.</p>
LogonID (UID)	<p>The default logon ID used to connect to your Sybase database. This ID is case-sensitive. A logon ID is required only if security is enabled on your database. If so, contact your system administrator to get your logon ID.</p>
NetworkAddress (NA)	<p>The network address depends on which network protocol is chosen under Network Library Name and on the Sybase server. If you have chosen Winsock, specify an IP address as follows: "<i>servername or IP address, port number</i>". For example, if your network supports named servers, you may specify an address such as "Sybaseserver, 5000". You may also specify the IP address directly such as "199.226.224.34, 5000".</p> <p>If you have chosen NamedPipes as the network protocol, you must specify the pipe address of the server. For example, "\\machine1\sybase\pipe\query".</p>
NetworkLibrary Name (NLM)	<p>NetworkLibraryName={Winsock   NamedPipes}. The name of the network library. This specifies which network protocol to use.</p> <p>The initial default is Winsock.</p> <p>This option has no effect on UNIX; on UNIX, TCP/IP is used.</p>




**Table 15-1. Sybase ASE Connection String Attributes** (cont.)

Attribute	Description
OptimizePrepare (OP)	<p data-bbox="471 295 1270 364">OptimizePrepare={0   1   2   3}. Determines whether stored procedures are created on the server for calls to SQLPrepare.</p> <p data-bbox="471 364 1270 486">When set to 0, stored procedures are created for every call to SQLPrepare. This setting can result in decreased performance when processing statements that do not contain parameters.</p> <p data-bbox="471 486 1270 624">When set to 1 (the initial default), the driver creates stored procedures only if the statement contains parameters. Otherwise, the statement is cached and run directly at the time of SQLExecute.</p> <p data-bbox="471 624 1270 763">When set to 2, 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.</p> <p data-bbox="471 763 1270 937">When set to 3, 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.</p>
PacketSize (PS)	<p data-bbox="471 937 1270 1076">PacketSize={-1   0   x}. Determines the number of bytes per network packet transferred from the database server to the client. The correct setting of this attribute can improve performance.</p> <p data-bbox="471 1076 1270 1180">When set to -1, the driver computes the maximum allowable packet size on the first connect to the data source and saves the value in the system information.</p> <p data-bbox="471 1180 1270 1249">When set to 0 (the initial default), the driver uses the default packet size as specified in the Sybase server configuration.</p> <p data-bbox="471 1249 1270 1392">When set to x, an integer from 1 to 1024, the driver uses a packet size represented by x times 512 bytes. For example, PacketSize=6 means to set the packet size to 6 * 512 bytes (3072 bytes).</p>

**Table 15-1. Sybase ASE Connection String Attributes** (cont.)

Attribute	Description
PacketSize (PS) (cont.)	<p>To take advantage of this connection attribute, you must configure the Sybase server for a maximum network packet size greater than or equal to the value you specified for PacketSize. For example:</p> <pre data-bbox="451 456 1143 552"> sp_configure "maximum network packet size", 5120 reconfigure Restart Sybase Server </pre> <p>NOTE: The ODBC specification specifies a connect option, SQL_PACKET_SIZE, that offers this same functionality. To avoid conflicts with applications that may set both the connection string attribute and the ODBC connect option, they have been defined as mutually exclusive. If PacketSize is specified, you will receive a message "Driver Not Capable" if you attempt to call SQL_PACKET_SIZE. If you do not set PacketSize, then application calls to SQL_PACKET_SIZE are accepted by the driver.</p>
Password (PWD)	A case-sensitive password.
RaiseErrorPosition Behavior (REPB)	<p>RaiseErrorPositionBehavior={0   1}. Specifies when the error is returned and where the cursor is positioned when raiserror is encountered.</p> <p>When set to 0 (the initial default), raiserror is handled separately from surrounding statements. The error is returned when raiserror is processed via SQLExecute, SQLExecDirect, or SQLMoreResults. The result set is empty.</p> <p>When set to 1 (MS compatible), raiserror is handled with the next statement. The error is returned when the next statement is processed; the cursor is positioned on the first row of subsequent result set. This could result in multiple raiserrors being returned on a single execute.</p>

**Table 15-1. Sybase ASE Connection String Attributes** (cont.)

Attribute	Description
SelectMethod (SM)	<p>SelectMethod={0   1}. Determines whether database cursors are used for Select statements.</p> <p>When set to 0 (the initial default), 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.</p> <p>When set to 1, Select statements are run directly without using database cursors, and the data source is limited to one active statement.</p>
TightlyCoupled DistributedTransactions (TCDT) 	<p>TightlyCoupledDistributedTransactions={0   1}. Determines whether the driver uses tightly coupled distributed transactions when connected to an Sybase ASE version 12 database. When set to 1 (the initial default), the driver uses this type of transaction and multiple connections within the same distributed transaction do not obey each other's locks.</p> <p>When set to 0, the overall performance of the driver is better, but multiple connections within the same distributed transaction may hang each other because the connections do not obey each other's locks.</p> <p>This attribute is valid only when the driver is enlisted in a distributed transaction or when it is connected to a Sybase ASE version 12 database. Otherwise, this attribute is ignored.</p>
WorkstationID (WKID)	The workstation ID used by the client.

---

## Data Types

[Table 15-2](#) shows how the Sybase ASE data types are mapped to the standard ODBC data types.

---

**Table 15-2. Sybase ASE Data Types**

---

<b>Sybase ASE</b>	<b>ODBC</b>
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR
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
timestamp	SQL_VARBINARY
tinyint	SQL_TINYINT
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

---

---

## MTS Support



On Windows, the Sybase ASE driver can take advantage of Microsoft Transaction Server (MTS) capabilities, specifically, the Distributed Transaction Coordinator (DTC). Refer to the help file of the "MicroSoft Transaction Server SDK" for details.

You can choose between Native OLE and XA protocol distributed transactions by using the `DistributedTransactionModel` connection string attribute documented in [Table 15-1 on page 320](#).

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.

---

## Unicode Support

The Sybase ASE driver supports Unicode if the Sybase ASE data source you are using to connect to the database is configured to use the UTF-8 character set. The data source is configured to use the UTF-8 character set by either setting the `Charset` attribute to UTF-8 in the connection string or in the `odbc.ini` file. See [Table 15-1 on page 320](#) for more information about the `Charset` attribute.

If the UTF-8 character set is installed, the driver maps the Sybase ASE data types as follows:

<b>Sybase ASE Data Type</b>	<b>Mapped to . . .</b>
Char	SQL_WCHAR
Varchar	SQL_WVARCHAR
Text	SQL_WLONGVARCHAR

This driver supports the Unicode ODBC function calls, called W (Wide) calls (for example, SQLConnectW). These calls are used to accept Unicode datastreams.

## Default Unicode Mapping

The default Unicode mapping for an application's SQL\_C\_WCHAR variable is:

<b>Platform</b>	<b>Default Unicode Mapping</b>
Windows	UCS-2
AIX	UTF-8
HP-UX	UTF-8
Solaris	UTF-8
Linux	UTF-8

## Connection Attributes for Unicode

Two new connection attributes are available to support Unicode. These attributes determine how character data is converted and



presented to an application and the database. The connection attributes are:

SQL_ATTR_APP_WCHAR_TYPE (1061)	Sets the SQL_C_WCHAR type for parameter and column binding to the desired unicode type, either SQL_DD_CP_UCS2 or SQL_DD_CP_UTF8. The default is the default Unicode mapping (see the previous section, " <a href="#">Default Unicode Mapping</a> ").
SQL_ATTR_DBMS_CODE_PAGE (1062)	Sets the code page type of the database. The main purpose of this attribute is to set the code page type of a database for drivers that cannot determine the database's code page. The default is SQL_DD_CP_ANSI.

Valid values for the two connection attributes are:

- SQL\_DD\_CP\_ANSI (ANSI code page)
- SQL\_DD\_CP\_UCS2 (UCS-2 code page)
- SQL\_DD\_CP\_UTF8 (UTF-8 code page)

You can set these connection attributes either before or after a connection is made. If the connection attributes are changed after a connection is established, all conversions use the new values.

The driver does not verify that the connection attributes are set only once per connection. Conversions are made based on the current application and database settings.

If the application does **not** set the SQL\_ATTR\_DBMS\_CODE\_PAGE attribute, the driver tries to determine the database's code page type; if the driver cannot determine the code page type, the

driver sets the `SQL_ATTR_DBMS_CODE_PAGE` attribute to `SQL_DD_CP_ANSI`. If the application does set the `SQL_ATTR_DBMS_CODE_PAGE` attribute, the driver, even if it can determine the database's code page type, does not override the value of the attribute set by the application.

If a driver does not support Unicode, `SQLGetConnectAttr` and `SQLSetConnectAttr` return `HYC00`.

These new connection attributes and their valid values can be found in the file `qesqlx.h`, which is installed with the driver.

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on `STATIC` cursors. For example:

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

- 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 regarding Step 1:** 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.
```

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, `SQL_PERSIST_AS_XML`. A client application must call `SQLSetStmtAttr` with this new attribute as an argument. See the following table for the definition of valid arguments for `SQLSetStmtAttr`.

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

---

## Support for Query Timeout



The Sybase ASE driver supports the `QUERY_TIMEOUT` statement attribute on Windows only.

---

## Isolation and Lock Levels Supported

Sybase ASE supports isolation levels 0 (read uncommitted), 1 (read committed, the default), 2 (repeatable read), and 3 (serializable). It supports page-level locking. See [Appendix D, “Locking and Isolation Levels”](#) on page 439 for details.

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the Sybase ASE driver. In addition, the following functions are supported:

- SQLColumnPrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures
- SQLTablePrivileges

The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

The Sybase database system supports multiple connections and multiple statements per connection. If `SelectMethod=1`, Sybase data sources are limited to one active statement in manual commit mode.

# 16 Connect ODBC for Text

Connect ODBC for Text (the "Text driver") supports ASCII text files in the Windows and UNIX environments. 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.

See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows and UNIX environments supported by this driver.

See the README file shipped with your DataDirect product for the file name of the text 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.

---

## 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 16-1](#).

---

**Table 16-1. Common Text File Formats**

---

<b>Format</b>	<b>Description</b>
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

On Windows, data sources are configured and modified through the ODBC Administrator.



**NOTE:** In the UNIX environment, there is no ODBC Administrator. To configure a data source in the UNIX environment, you must edit the system information file using the attributes in [Table 16-4 on page 352](#). You must also edit this file to perform a translation. See [Appendix H, “The UNIX Environments” on page 473](#) for information about editing the file.

## To configure a Text data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the ODBC Text Driver Setup dialog box.

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



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

**Data Source Name:** Type a string that identifies this Text data source configuration in the system information. Examples include "Accounting" or "Text Files."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Files" or "My Text Files in the Accounting Directory."



**Database Directory:** Type the directory in which the text files are stored. If none is specified, the current working directory is used.

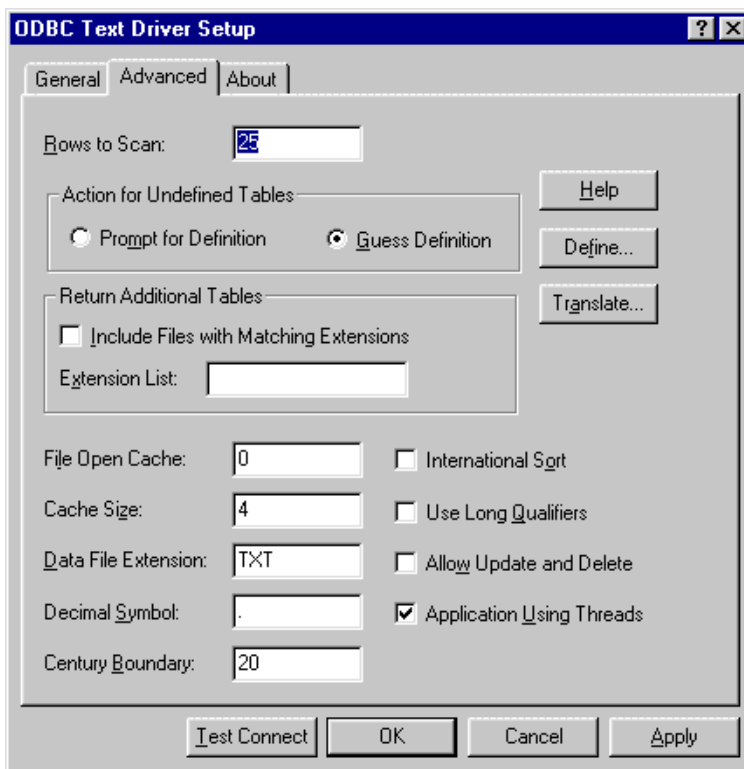
**Default Table Type:** Select the type of text file: comma-separated, tab-separated, character-separated, fixed length, or stream. This value tells the driver the default type, which is used when creating a new table and opening an undefined table.

**Delimiter Character:** Type the character used as a delimiter for character-separated files. It can be any printable character except for single and double quotes. The default is a comma (,).

**Column Names in First Line:** Select this check box to tell the driver to look for column names in the first line of the file.

NOTE: The Default Table Type, Delimiter Character, and Column Names in First Line settings apply only to tables not previously defined. These fields also determine the attributes of new tables created with the Create Table statement.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Rows to Scan:** Type the number of rows in a text file that the driver scans to determine the data types in the file. If the value is 0, all rows in the file are scanned. The default is 25.

**Action for Undefined Tables:** Select one of the two options in this group to indicate which action the driver should take when it encounters a file that has not been defined. Select the Prompt for Definition option if you want the driver to prompt the user when it encounters a file whose format is not defined. Otherwise, select the Guess Definition option; in this case, the driver guesses the file's format.

**Include Files with Matching Extensions:** Select this check box to tell the driver to return files with a given extension in addition to the files specified in the Data File Extension field.

**Extension List:** If you selected the Include Files with Matching Extensions check box, type a comma-separated list of extensions of files you want returned. 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.

**File Open Cache:** Type an integer value that specifies the maximum number of unused file opens to cache. For example, the value 4 specifies that when a user opens and closes four 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 degrades performance. The advantage of file open caching is increased performance. The disadvantage is that a user who specifies file locking on open may get a locking conflict even though no one appears to have the file open. The default is 0, which means no file open caching.

**Cache Size:** Type the number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 cannot see updates made by other users until you run the Select statement again. The default is 4.

**Data File Extension:** Type the file extension to use for data files. The default value is TXT. This setting cannot exceed three characters. The Data File Extension setting is used for all Create Table statements. Sending a Create Table using an extension other than the Data File Extension setting causes an error.

In other SQL statements, such as Select or Insert, users can specify an extension other than the Data File Extension setting. The Data File Extension setting is used when no extension is specified.

**Decimal Symbol:** Type the decimal separator used when data is stored. Valid values are a comma or a period. The international decimal symbol (.), which is the default, must be used in DML statements and parameter buffers.

**Century Boundary:** Type the cutoff year for century inference when converting two-digit dates to four-digit dates. Two-digit dates that are less than the specified year number will be converted to 20xx. Two-digit dates greater than or equal to the number are converted to 19xx. The default value is 20. For example, using the default value, a date of 19 will be interpreted as 2019 and a date of 21 is interpreted as 1921.

**International Sort:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Do not select this check box if you want to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.



**Use Long Qualifiers (Windows only):** Select this check box to use long path names as table qualifiers. When you select this check box, path names can be up to 255 characters. The default length for path names is 128 characters.

**Allow Update and Delete:** Select this check box to allow Update and Delete statements. Because Update and Delete

statements cause immediate changes to a table, only one connection at a time can operate on a table. When this check box is selected, tables are opened exclusively by the current connection. 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.

**Application Using Threads:** Select this check box to ensure that the driver works with multi-threaded applications. Do not select this check box when you are using the driver with single-threaded applications. Turning off this setting avoids additional processing required for ODBC thread-safety standards.

**Define:** Click **Define** to define the structure of your text files as described in [“Defining Table Structure”](#) on page 342.

**Translate:** Click **Translate** to display the Select Translator dialog box, which lists the translators specified in the ODBC Translators section of the system information. 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 and perform the translation.

- 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 improper environment or incorrect connection value, it will display 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.

---

## Defining Table Structure

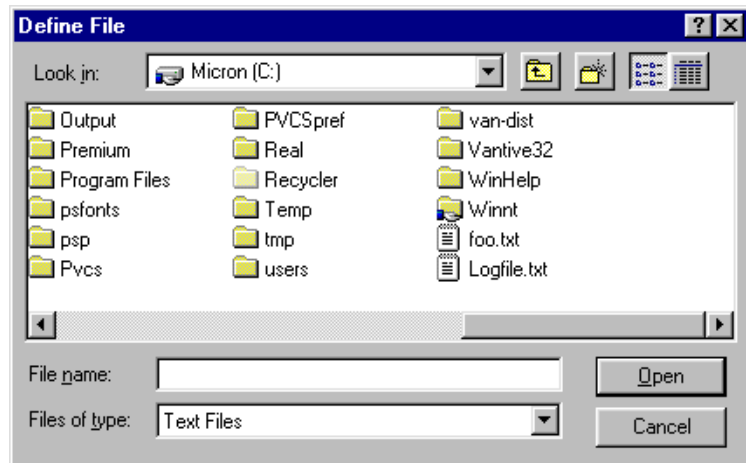


Note that this section does not apply to the UNIX platforms. See [“Defining Table Structure on UNIX Platforms” on page 346](#) for information on how to define table structure on the UNIX platforms.

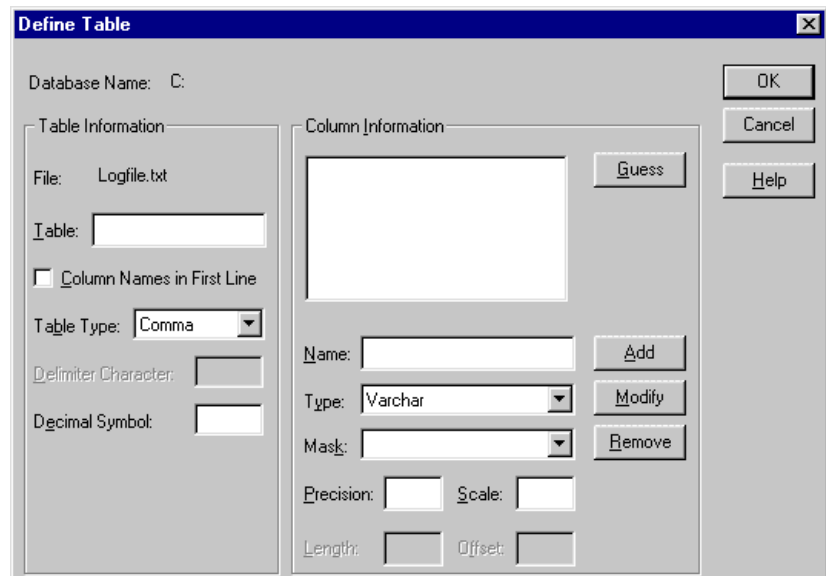
Because text files do not all have the same structure, the driver provides the option of defining the structure of an existing file. Although defining the structure is not mandatory (the driver can attempt to guess the names and types of the columns), this feature is extremely useful.

Define the structure of a file as follows:

- 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. The name may be up to 32 characters in length 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.

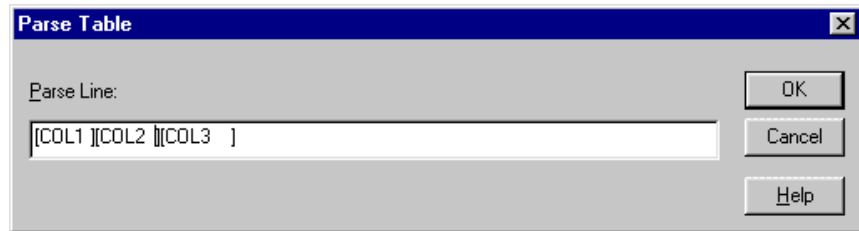
**Delimiter Character:** If the table type is Character, type the delimiter used in character-separated files. The value can be any printable character except single and double quotes.

**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** button is active and you can click it 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 **Parse** button is active and you can click it to 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 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, you must select a date mask for the field or type one in. See ["Date Masks" on page 349](#) 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; for binary data types, precision is defined as the length in bytes of the data. For time, timestamp, and all interval data types, precision is the number of characters in the character representation of this 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 decimal and numeric data types is defined as the maximum number of digits to the right of the decimal point. For approximate floating point number columns, the scale is undefined, since the number of digits to the right of the decimal point is not

fixed. For datetime or interval data that contains a seconds component, the scale is defined as the number of digits to the right of the decimal point in the seconds component of the data. 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 Platforms



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 system information file (odbc.ini) or in the connection string.

Define the QETXT.INI file as follows:

- 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:

```
emptxt.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=emptxt.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 349](#) 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 16-2](#) lists the symbols to use when specifying the date mask.

**Table 16-2. 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 (that is, jan, Jan, JAN).
mmmm, Mmmm, MMMM	Output the full month name depending on the case of the Ms (that is, 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 (that is, mon, Mon, MON).
dddd, Dddd, DDDD	Output the day depending on the case of the Ds (that is, 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.

**Table 16-2. Date Masks for Text Driver** (cont.)

Symbol	Description
\ - . : , (space)	Special characters used to separate the parts of a date.
\	Output the next character. For example, if the mask is mm/dd/yyyy \AD, the value appears as 10/01/1993 AD in the text file.
"string", 'string'	Output the string in the text file.

[Table 16-3](#) shows some example date values, masks, and how the date appears in the text file.

**Table 16-3. Date Mask Examples**

Date	Mask	Value
1993-10-01	yyyy-mm-dd	1993-10-01
	m/d/yy	10/1/93
	Ddd, Mmm dd, yyyy	Fri, Oct 01, 1993

---

# Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information file (odbc.ini). These values are not written to the system information file.

You can specify either long or short names in the connection string. The connection string has the form:

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

An example of a connection string for text files is:

```
DSN=TEXT FILES;TT=CHARACTER;DC=&
```

[Table 16-4](#) gives the long and short names for each attribute, as well as a description.



To configure a data source in the UNIX environment, you must edit the system information file. This file accepts only long names for attributes. See [Appendix H, “The UNIX Environments” on page 473](#) for information about this file.

[Table 16-4](#) also lists the initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.

**Table 16-4. Text Connection String Attributes**

Attribute	Description
AllowUpdateAndDelete (AUD)	<p>AllowUpdateAndDelete={0   1}. Determines whether a data source allows Update and Delete statements. Because Update and Delete statements cause immediate changes to a table, only one connection at a time can operate on a table.</p> <p>When set to 1, tables are opened exclusively by the current connection. Each update and delete on a text file can cause significant changes to the file, and performance may be poor. Consider a more appropriate database form if performance is a significant factor.</p> <p>The initial default is 0.</p>
AppCodePage (ACP)	<p>Valid values for this attribute are listed in <a href="#">Appendix I</a>. The code page that you specify must be the same as the code page used by your application.</p> <p>The driver on UNIX determines the value of the application's code page by checking for an AppCodePage value in the following order:</p> <ul style="list-style-type: none"> <li>■ In the connection string</li> <li>■ In the DataSource section of the system file (odbc.ini)</li> <li>■ In the ODBC section of the system file (odbc.ini)</li> </ul> <p>If no AppCodePage value is found, the driver uses the default value of 1 (ISO 8859-1 Latin-1).</p>
ApplicationUsingThreads (AUT)	<p>ApplicationUsingThreads={0   1}. Ensures that the driver works with multi-threaded applications.</p> <p>When set to 1 (the initial default), the driver is thread-safe.</p> <p>When using the driver with single-threaded applications, you may set this option to 0 to avoid additional processing required for ODBC thread-safety standards.</p>

NOTE: The ScanRows, TableType, Delimiter, and FirstLineNames attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement.



**Table 16-4. Text Connection String Attributes** (cont.)

<b>Attribute</b>	<b>Description</b>
CacheSize (CSZ)	<p>The number of 64 KB blocks the driver uses to cache database records. The greater the number of blocks, the better the performance. 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 will not be able to see updates made by other users until you run the Select statement again.</p> <p>The initial default is 4.</p>
CenturyBoundary (CB)	<p>CenturyBoundary=20. Specifies the cutoff year for century inference when converting two-digit dates to four-digit dates. Two-digit dates that are less than the specified year number will be converted to 20xx. Two-digit dates greater than or equal to the number will be converted to 19xx. The default value is 20. For example, using the default value, a date of 19 will be interpreted as 2019 and a date of 21 will be interpreted as 1921.</p>
Database (DB)	<p>The directory in which the text files are stored.</p>
DataFileExtension (DFE)	<p>The file extension to use for data files. The Data File Extension setting cannot be greater than three characters. The Data File Extension setting is used for all Create Table statements. Sending a Create Table using an extension other than the Data File Extension setting causes an error.</p> <p>In other SQL statements, such as Select or Insert, users can specify an extension other than the Data File Extension setting. The Data File Extension setting is used when no extension is specified.</p> <p>The initial default is TXT.</p>
DataSourceName (DSN)	<p>A string that identifies a Text data source configuration in the system information. Examples include "Accounting" or "Text Files."</p>
DecimalSymbol (DS)	<p>DecimalSymbol={,   ,}. Specifies the decimal separator used when data is stored. The international decimal symbol (.), which is the initial default, must be used in DML statements and parameter buffers.</p>

NOTE: The ScanRows, TableType, Delimiter, and FirstLineNames attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement.

**Table 16-4. Text Connection String Attributes** (cont.)

Attribute	Description
Delimiter (DC)	The character used as a delimiter for character-separated files. It can be 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 ODBC Administrator (Windows only). The initial default is a comma (,).
ExtraExtensions (EE)	A list of additional file name extensions to be recognized as text tables. When an application requests a list of tables, only files that have been defined are returned. To have the driver also return names of undefined files, specify a comma-separated list of file extensions. To specify files with no extension, use the keyword <code>None</code> .
FileOpenCache (FOC)	The maximum number of unused file opens to cache. For example, when <code>FileOpenCache=4</code> , and a user opens and closes four files, the files are not actually closed. The driver keeps them open so that if another query uses one of these files, the driver does not have to perform another open, which is expensive. The advantage of using file open caching is increased performance. The disadvantage is that a user who tries to open the file exclusively may get a locking conflict even though no one appears to have the file open. The initial default is 0.
FirstLineNames (FLN)	<code>FirstLineNames={0   1}</code> . Determines whether the driver looks for column names in the first line of the file. When set to 0 (the initial default), the first line is interpreted as the first record in the file. When set to 1, the driver looks for column names in the first line of the file.

NOTE: The `ScanRows`, `TableType`, `Delimiter`, and `FirstLineNames` attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the `Create Table` statement.

**Table 16-4. Text Connection String Attributes** (cont.)

Attribute	Description
IntlSort (IS)	<p data-bbox="471 303 1272 407">IntlSort={0   1}. Determines the order in which records are retrieved when you issue a Select statement with an Order By clause.</p> <p data-bbox="471 416 1272 546">When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p data-bbox="471 555 1272 720">When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>
ScanRows (SR)	<p data-bbox="471 729 1272 824">The number of rows in a text file that the driver scans to determine the column types in the file. If the value is 0, all rows in the file are scanned.</p> <p data-bbox="471 833 771 868">The initial default is 25.</p>
TableType (TT)	<p data-bbox="471 876 1272 1039">TableType={Comma   Tab   Character   Fixed   Stream}. The Text driver supports five table types: comma-separated, tab-separated, character-separated, fixed length, and stream. Setting this value tells the driver the default type, which is used when creating a new table and opening an undefined table.</p>

---

NOTE: The ScanRows, TableType, Delimiter, and FirstLineNames attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement.

**Table 16-4. Text Connection String Attributes** (cont.)

Attribute	Description
UndefinedTable (UT)	<p>UndefinedTable={PROMPT   GUESS}. Determines whether the driver should prompt the user when it encounters a table for which it has no structure information.</p> <p>When set to PROMPT, the driver is told to display a dialog box that allows the user to describe the file's format.</p> <p>When set to GUESS (the initial default), the driver is told to guess the file's format.</p>
UseLongQualifiers (ULQ)	<p>UseLongQualifiers={0   1}. Determines whether the driver uses long path names as table qualifiers.</p> <p>When set to 0 (the initial default), the driver does not use long path names (the maximum path name length is 128 characters).</p> <p>When set to 1, the driver uses long path names (the maximum path name length is 255 characters).</p>

NOTE: The ScanRows, TableType, Delimiter, and FirstLineNames attributes apply to tables that have *not* been defined. These attributes also determine the characteristics of new tables created with the Create Table statement.

## Data Types

Table 16-5 shows how the text file data types are mapped to the standard ODBC data types.

**Table 16-5. Text Data Types**

Text	ODBC
Numeric	SQL_NUMERIC
Date	SQL_TYPE_DATE
Varchar	SQL_VARCHAR

---

## Select Statement

You use the SQL Select statement to specify the columns and records to be read. All Select statement clauses described in [Appendix A, “SQL for Flat-File Drivers” on page 393](#) are supported by the 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

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the Text driver. In addition, the following function is supported: SQLSetPos.

The Text driver also supports backward and random fetching in SQLExtendedFetch and SQLFetchScroll. The driver supports the minimum SQL grammar.

---

## Number of Connections and Statements Supported

Text files support multiple connections and multiple statements per connection.

# 17 Connect ODBC for XML

Connect ODBC for XML (the "XML driver") can create and read Microsoft Data Islands, ADO 2.5 persisted files, and DataDirect formatted files. See ["Supported Formats for XML Files" on page 361](#) for more details.

In addition, the driver can read additional XML formats with help from hints that you specify. See ["Using Hints" on page 372](#) for details.

The XML driver runs in the Windows environments. See ["Environment-Specific Information" on page 33](#) for detailed information about the Windows environments supported by this driver.

The XML driver includes a SQL Engine that provides ANSI SQL-92 support. The SQL Engine included with the XML driver supports the following SQL statements:

- Select
- Create and Drop Table
- Insert
- Update
- Delete

See ["SQL Supported" on page 381](#) for more information about the SQL supported.

See the README file shipped with your DataDirect product for the file name of the XML driver.

---

## Driver Requirements

You must have Internet Explorer 5 installed.

---

## Terminology

Connect for XML uses some terms differently than other drivers and databases. The following table identifies some of these differences.

Catalog	File system directory or URL.
Row	<p>The &lt;row&gt; element in an ADO 2.5 persisted XML file or the &lt;record&gt; element in the DataDirect persisted XML format.</p> <p>The row element in the Data Island format has no defined name—it can be any name the user chooses.</p>
Schema	The XML driver recognizes the use of a document schema. A document schema can specify the set of allowable elements, their sequence, relationship to each other, and their cardinality. The XML driver does not use the term in the context of "logging on to a user ID."
Table	Set of sibling elements, each with the same set of child elements or attributes. A table is XML rectangular markup. This could be Microsoft Data Islands, ADO 2.5 persisted files, or DataDirect format. See <a href="#">"Supported Formats for XML Files"</a> on page 361 for a description of these different formats.



---

## Supported Formats for XML Files

The XML driver can create and read the XML formats described in [Table 17-1](#). The driver can read additional XML formats with help from hints that you specify. See [“Using Hints” on page 372](#) for details.

---

**Table 17-1. Common XML File Formats**

---

Format	Description
ADO 2.5 persisted files	<p>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:</p> <pre>&lt;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"&gt;</pre>
DataDirect Format	<p>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):</p> <pre>&lt;table 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"&gt;</pre>
Microsoft Data Islands	<p>These are identified by the &lt;XML&gt; 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:</p> <pre>&lt;Schema xmlns="urn:schemas-microsoft-com:xml-data"       xmlns:dt="urn:schemas-microsoft-com:datatypes"&gt;</pre>

---

---

## Configuring Data Sources

To configure an XML data source:

- 1 Start the ODBC Administrator to display a list of data sources.
- 2 If you are configuring an existing data source, select the data source name and click **Configure** to display the DataDirect Connect for XML Setup dialog box.

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



**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.

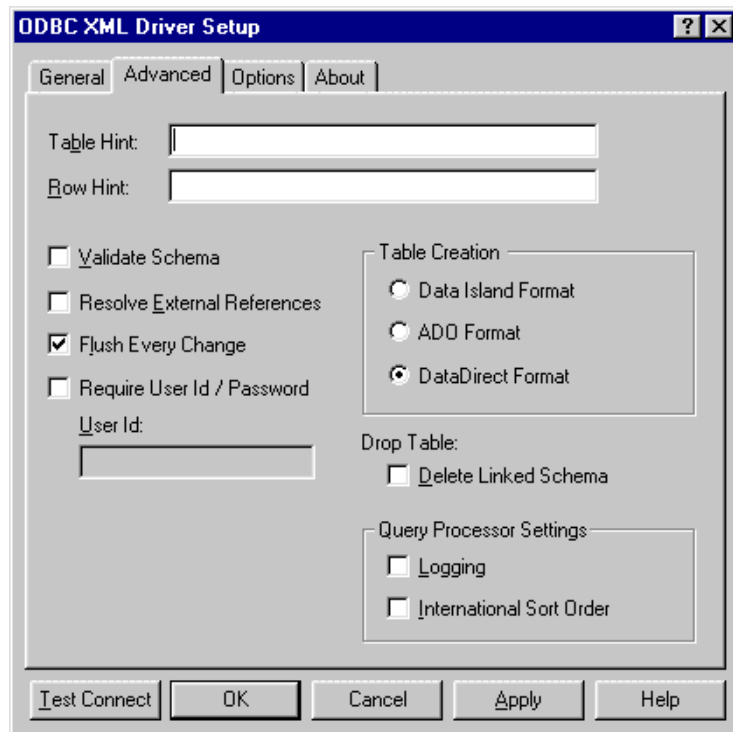
- 3 On the General tab, provide the following information; then, click **Apply**.

**Data Source Name:** Type a string that identifies this XML data source configuration in the system information. Examples include "Accounting" or "XML Files."

**Description:** Type an optional long description of a data source name. For example, "My Accounting Files" or "My XML Files in the Accounting Directory."

**Location:** Type a directory or URL that contains the XML files. Or, browse for the directory by clicking the button next to the Location field to open the Find Directory window.

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



On this tab, provide any of the following optional information; then, click **Apply**.

**Table Hint:** Type a string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the table or rowset nodes in an XML file. See [“Using Hints” on page 372](#) for details.

**Row Hint:** Type a string that specifies an XSL pattern to identify the nodes that make up the rows in the rowset. See [“Using Hints” on page 372](#) for details.

**Validate Schema:** Select this check box to enable the validation of the XML file against its schema. By default, this check box is not selected. To process an XML document even if the document is not valid, clear the check box.

**Resolve External References:** Select this check box to enable the resolution of external references, such as DTDs, Schemas, Entities, and Notations. By default, this check box is not selected. Clearing this box allows the document to be processed, even if the XML parser cannot locate the external references.

**Flush Every Change:** Select this check box to write the document to disk after every change. By default, this check box is selected. When this check box is not selected, the driver does not write the document to disk after every insert, update, or delete operation. Clearing this check box can speed up performance.

**Require User ID/Password:** Select this check box to specify whether a User ID and password are required to establish a connection to the data source. By default, this check box is not selected. When this check box is not selected, no user ID and password are required to establish a connection to the data source.

**User ID:** Type the default user ID used to establish a connection to the data source. This field is enabled when the Use Required User ID/Password check box is selected.

**Table Creation:** Select one of these options to determine the style of XML that is generated when a new table is created. By default, DataDirect Format is selected.

- **Data Island Format:** This option causes new tables to be created with the IE 5 Data Island XML style.
- **ADO Format:** This option causes new tables to be created with the ADO 2.5 XML style.
- **DataDirect Format:** This option causes new tables to be created with the DataDirect format. This format complies to the W3C recommendation for XML schema, working draft April 07, 2000.

**Delete Linked Schema:** Select this check box to specify whether externally linked schema files are deleted when a table is deleted. The XML file for the table contains a link to this external schema file. If multiple XML files are linked to the same schema file, the schema file should not be deleted when a table is deleted. By default, this check box is not selected.

**Logging:** Select this check box to control whether logging is enabled. The log file logs the SQL execution plan. By default, this check box is not selected. If you select the check box, a log file is created in the current directory. The default log file name is \Integrator.txt.

**International Sort Order:** Select this check box to indicate the order in which records are retrieved when you issue a Select statement with an Order By clause. Clear this box to use ASCII sort order (the default setting). This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."

Select this check box to use international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating

system documentation about the sorting of accented characters.

- 5 Optionally, click the **Options** tab to specify data source connection values.



**Driver Options:** Type configuration options specific to the XML driver.

---

**WARNING:** The properties you set in the Options tab override other properties for this session only and can adversely affect the operation of the data provider. Use only authorized entries. For information about authorized entries for the Options tab, contact MERANT technical support.

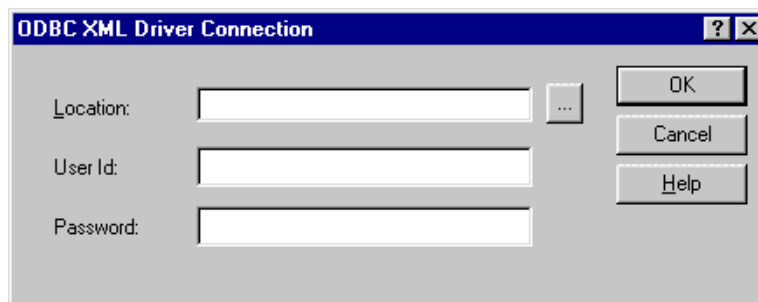
---

- 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 Setup window.
  - 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 will display 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.

---

## Connecting to a Data Source 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:



In this dialog box, perform the following steps:

- 1 Type the URL or directory location that contains the XML files into the entry field. UNC paths are supported.

To browse for the directory, click the button next to the Location field.

- 2 If required, type your user ID and password in the appropriate fields.
- 3 Click **OK** to connect to the data source.

---

## Connecting to a Data Source Using a Connection String

If your application requires a connection string to connect to a data source, you must specify the data source name that tells the driver which section in the system information to use for the default connection information. Optionally, you may specify *attribute=value* pairs in the connection string to override the default values stored in the system information. These values are not written to the system information.

The connection string has the form:

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

An example of a connection string for XML files is:

```
DSN=XML FILES;Create Type=ADO25
```

**Table 17-2** gives the names and descriptions of the attributes. It also lists the initial defaults that apply when no value is specified in either the connection string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source, that value is the default.



**Table 17-2. XML Connection String Attributes**

<b>Attribute</b>	<b>Description</b>
Create Type	<p>Create Type={IE5DataIsland   ADO25   DataDirect}. Determines the style of XML that is generated when a new table is created.</p> <p>When set to IE5DataIsland, new tables are created with the Internet Explorer 5 Data Island XML style.</p> <p>When set to ADO25, new tables are created with the ADO 2.5 XML style.</p> <p>When set to DataDirect (the initial default), new tables are created with the DataDirect format. This format complies to the W3C recommendation for XML schema, working draft April 07, 2000.</p>
Delete Schema	<p>Delete Schema={0   1}. Determines whether externally linked schema files are deleted when a table is deleted. The XML file for the table contains a link to this external schema file. If multiple XML files are linked to the same schema file, the schema file should not be deleted when a table is deleted.</p> <p>When set to 0 (the initial default), externally linked schema files are not deleted when a table is deleted.</p> <p>When set to 1, externally linked schema files are deleted when a table is deleted.</p>
International Sort	<p>International Sort={0   1}. Determines the order that records are retrieved when you issue a Select statement with an Order By clause.</p> <p>When set to 0 (the initial default), the driver uses the ASCII sort order. This order sorts items alphabetically with uppercase letters preceding lowercase letters. For example, "A, b, C" would be sorted as "A, C, b."</p> <p>When set to 1, the driver uses the international sort order as defined by your operating system. This order is always alphabetic, regardless of case; the letters from the previous example would be sorted as "A, b, C." See your operating system documentation concerning the sorting of accented characters.</p>
Log Filename	<p>The name of the file in which logging will occur. The initial default log file name is \Integrator.txt.</p>

**Table 17-2. XML Connection String Attributes** (cont.)

Attribute	Description
Logging	<p>Logging={0   1}. Determines whether internal information is logged.</p> <p>When set to 0 (the initial default), internal information is logged.</p> <p>When set to 1, internal information is <b>not</b> logged.</p>
Max BLOB Compare Size	<p>The number of bytes at the beginning of a BLOB when making comparisons in the XML driver's evaluator. If a BLOB is larger than specified length, it is truncated before comparison.</p>
Max BLOB Sort Size	<p>The number of bytes at the beginning of a BLOB that are used when performing an Order By, Distinct, or Group By on a BLOB column. This option can affect in-memory indexes that are built over BLOB columns during a join.</p> <p>The initial default is 512.</p> <p>NOTE: Do not set the value higher than the default if the client application will use BLOBs as join condition columns.</p>
Password	<p>The password that the application uses to connect to your XML database. A password is required only if security is enabled on your database. If so, contact your system administrator to get your password.</p>
Read Only	<p>Read Only={0   1}. Determines whether the data source is read-only.</p> <p>When set to 0, the data source is configured to be read/write (the initial default).</p> <p>When set to 1, the data source is configured to be read-only.</p>
Require Passwd	<p>Require Passwd={0   1}. Specifies whether a User ID and password are required to establish a connection to the data source.</p> <p>When set to 0 (the initial default), a User ID and password are not required.</p> <p>When set to 1, a User ID and password are required.</p>

**Table 17-2. XML Connection String Attributes** (cont.)

<b>Attribute</b>	<b>Description</b>
Resolve External	<p>Resolve External={0   1}. Determines whether external references such as DTDs, Schemas, Entities, and Notations are resolved.</p> <p>When set to 0, the document is allowed to be processed, even if the XML parser cannot locate the external references.</p> <p>When set to 1 (the initial default), the document is not allowed to be processed if the XML parser cannot locate the external references.</p>
Row Hint	<p>A string that specifies an Extensible Stylesheet Language (XSL) pattern to identify the nodes that make up the rows in the rowset. See <a href="#">"Using Hints" on page 372</a> for details.</p>
Table Hint	<p>A string that specifies an XSL pattern to identify the table or rowset nodes in an XML file. See <a href="#">"Using Hints" on page 372</a> for details.</p>
Use Floating Point Rounding	<p>Use Floating Point Rounding={0   1}. Determines whether the XML evaluator performs a small amount of rounding when comparing floating-point numbers. The binary representation of two identical floating-point numbers can be slightly different.</p> <p>When set to 0 (the initial default), the XML evaluator does not perform a small amount of rounding when comparing floating-point numbers.</p> <p>When set to 1, the evaluator compensates for slight differences.</p>
User ID	<p>The User ID (user name) that the application uses to connect to your XML database. A User ID is required only if security is enabled on your database. If so, contact your system administrator to get your User ID.</p>
Validate Schema	<p>Validate Schema={0   1}. Determines whether the XML file is validated against its schema.</p> <p>When set to 0 (the initial default), the XML file is not validated against its schema.</p> <p>When set to 1, the XML file is validated against its schema. This allows a well-formed XML document to be processed, even if the document is not valid.</p>
XML.Initial Catalog	<p>An attribute that specifies the name of an initial directory or URL that contains XML or HTML files. When defined, schema rowsets look in this location to find tables.</p>

---

## Using Hints

The XML driver supports table and row hints. You can specify a table hint, a row hint, or both, on the Advanced tab of the ODBC XML Driver Setup dialog box.

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 XML format map to the standard ODBC data types, as follows:

- [Table 17-3. Data Islands Data Types](#)
- [Table 17-4. ADO 2.5 Persisted Files Data Types](#)
- [Table 17-5. DataDirect Data Types](#)

**Table 17-3. Data Islands Data Types**

<b>Data Islands</b>	<b>Internal XML Name</b>	<b>ODBC</b>
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_LONGVARCHAR
time	time	SQL_TYPE_TIME
ui1	ui1	SQL_TINYINT UNSIGNED
ui2	ui2	SQL_SMALLINT UNSIGNED
ui4	ui4	SQL_INTEGER UNSIGNED



**Table 17-4. ADO 2.5 Persisted Files Data Types**

<b>ADO 2.5 Persisted Files</b>	<b>Internal XML Name</b>	<b>ODBC</b>
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
wvarchar	string	SQL_VARCHAR
wlvarchar	string	SQL_LONGVARBINARY

**Table 17-5. DataDirect Data Types**

<b>DataDirect</b>	<b>Internal XML Name</b>	<b>ODBC</b>
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
wvarchar	string	SQL_VARCHAR
wlvarchar	string	SQL_LONGVARBINARY

---

## Persisting a Result Set as an XML Data File

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

- 1 Turn on STATIC cursors. For example:

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

- 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 regarding Step 1:** 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.

**Note regarding Step 3:** A new statement attribute is available to support XML persistence, SQL\_PERSIST\_AS\_XML. A client application must call SQLSetStmtAttr with this new attribute as an argument. See the following table for the definition of valid arguments for SQLSetStmtAttr.

Argument	Definition
<i>StatementHandle</i>	The handle of the statement that contains the result set to persist as XML.

Argument	Definition
<i>Attribute</i>	SQL_PERSIST_AS_XML. This new statement attribute can be found in the file qesqltext.h, which is installed with the driver.
<i>ValuePtr</i>	Pointer to a URL that specifies the full path name of the XML data file to be generated. The directory specified in the path name must exist, and if the specified file name exists, the file will be overwritten.
<i>StringLength</i>	The length of the string pointed to by ValuePtr or SQL_NTS if ValuePtr points to a null terminated string.

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

Function Sequence Error

---

## ODBC Conformance Level

See [Appendix C, “ODBC API and Scalar Functions”](#) on page 427 for a list of the API functions supported by the XML driver. In addition, the following function is supported: SQLSetPos.

---

# 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 381](#)
- [“Extensions to SQL Standards” on page 382](#)
- [“Grammar Token Definitions” on page 382](#)

### SQL Statements

The SQL Engine included with the XML driver supports the following SQL statements:

- Select
- Create and Drop Table
- Insert
- Update
- Delete

## Extensions to SQL Standards

The XML driver uses SQL grammar that is compliant with entry level ANSI SQL 92. [Table 17-6](#) summarizes significant extensions to the grammar.

---

**Table 17-6. 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 383](#)
- [“Delimited Identifiers” on page 383](#)
- [“Integer Numbers” on page 384](#)
- [“Real Numbers” on page 384](#)
- [“Character String Literals” on page 384](#)
- [“GUID Literals” on page 385](#)
- [“Hex Literals” on page 385](#)
- [“Time and Date Literals” on page 385](#)
- [“SQL Operators and Symbols” on page 386](#)

- [“Keywords for the XML Driver” on page 386](#)
- [“SQL Comments” on page 391](#)

## ***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 a `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.

	<b>ODBC Syntax</b>	<b>ANSI SQL 92 Syntax</b>
<b>Date Literal</b>	{d '1999-09-19'}	date '1999-09-19'
<b>Time Literal</b>	{t '11:11:11.225'}	time '11:11:11.225'
<b>Timestamp Literal</b>	{ts '1999-09-19 11:11:11.225'}	timestamp '1999-09-19 11:11:11.225'
<b>Timestamp Literal</b>	{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***

<b>Symbol</b>	<b>Description</b>	<b>Symbol</b>	<b>Description</b>
':'	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 17-7](#) lists all of the keywords that are reserved for use in SQL statements or designated as potential future reserved words.

---

**Table 17-7. 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	CASCADE
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

**Table 17-7. Reserved Keywords** (cont.)

DATA	DATE	DAY
DAYOFMONTH	DAYOFWEEK	DEALLOCATE
DEC	DECIMAL	DECLARE
DEFAULT	DEFERRABLE	DEFERRED
DELETE	DEPTH	DESC
DESCRIBE	DESCRIPTOR	DIAGNOSTICS
DICTIONARY	DISCONNECT	DISTINCT
DOMAIN	DOUBLE	DROP
EACH	ELSE	ELSEIF
END	END_EXEC	EQUALS
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FLOOR	FOR	FOREIGN
FOUND	FROM	FULL
GENERAL	GET	GLOBAL
GO	GOTO	GRANT
GROUP	HAVING	HOUR
IDENTIFY	IF	IFNULL
IGNORE	IMMEDIATE	IN
INDEX	INFO	INDICATOR
INITIALLY	INNER	INPUT
INSENSITIVE	INSERT	INT
INTEGER	INTERSECT	INTERVAL
INTO	IS	ISOLATION
JOIN	KEY	LANGUAGE
LAST	LCASE	LEADING
LEAVE	LEFT	LENGTH
LESS	LEVEL	LIKE

**Table 17-7. Reserved Keywords** (cont.)

LIMIT	LOCAL	LOOP
LOWER	LTRIM	MATCH
MAX	MIN	MINUTE
MOD	MODIFY	MODULE
MONTH	NAMES	NATIONAL
NATURAL	NCHAR	NEW
NEXT	NO	NONE
NOT	NOW	NULL
NULLIF	NUMERIC	OBJECT
OCTET_LENGTH	OF	OFF
OID	OLD	ON
ONLY	OPEN	OPERATION
OPERATORS	OPTION	OR
ORDER	OTHERS	OUTER
OUTPUT	OVERLAPS	PAD
PARAMETERS	PARTIAL	PENDANT
POSITION	POWER	PRECISION
PREORDER	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVATE
PRIVILEGES	PROCEDURE	PROTECTED
PUBLIC	RCASE	READ
REAL	RECURSIVE	REF
REFERENCES	REFERENCING	RELATIVE
REMOVE	REPLACE	RESIGNAL
RESTRICT	RETURN	RETURNS
REVOKE	RIGHT	ROLE
ROLLBACK	ROUND	ROUTINE
ROW	ROWS	RTRIM
SAVEPOINT	SCHEMA	SCROLL
SEARCH	SECOND	SECTION

**Table 17-7. Reserved Keywords** (cont.)

---

SELECT	SENSITIVE	SEQUENCE
SESSION	SESSION_USER	SET
SIGNAL	SIMILAR	SIZE
SMALLINT	SOME	SPACE
SQL	SQLCODE	SQLERROR
SQLLEXCEPTION	SQLSTATE	SQLWARNING
STRUCTURE	SUBSTRING	SUM
SYSTEM_USER	TABLE	TEMPORARY
TEST	THEN	THERE
TIME	TIMESTAMP	TIMEZONE_HOUR
TIMEZONE_MINUTE	TO	TRAILING
TRANSACTION	TRANSLATE	TRANSLATION
TRIGGER	TRIM	TRUE
TYPE	UCASE	UNDER
UNION	UNIQUE	UNKNOWN
UPDATE	UPPER	USAGE
USER	USING	VALUE
VALUES	VARCHAR	VARIABLE
VARYING	VIEW	VIRTUAL
VISIBLE	WAIT	WHEN
WHENEVER	WHERE	WHILE
WITH	WITHOUT	WORK
WRITE	YEAR	ZONE

---

## ***SQL Comments***

ANSI SQL 92 standard comments (--) and C++ standard comments (/\*...\*/, //) are supported. Comments can be nested.

For example, in the following query:

```
SELECT col1 /* col1 comment */
/*
    col2,    -- col2 comment
    col3,    // col3 comment
    col4,    /* col4 comment */
*/
FROM t1
```

columns col2, col3, and col4 will be ignored.





# A SQL for Flat-File Drivers

This appendix describes the SQL statements that you can use with the flat-file drivers (Btrieve, dBASE, Paradox, and Text). The Excel Workbook driver (also a flat-file driver) supports only the Select statement. Any exceptions to the supported SQL functionality described in this appendix is documented in the individual flat-file driver chapter.

The database drivers parse SQL statements and translate them into a form that the database can understand. The SQL statements discussed in this appendix let you:

- Read, insert, update, and delete records from a database
- Create new tables
- Drop existing tables

These SQL statements allow your application to be portable across other databases.

---

## Select Statement

The form of the Select statement supported by the flat-file drivers is:

```
SELECT [DISTINCT] { * | column_expression, ... }
FROM table_names [table_alias] ...
[ WHERE expr1 rel_operator expr2 ]
[ GROUP BY { column_expression, ... } ]
[ HAVING expr1 rel_operator expr2 ]
[ UNION [ALL] (SELECT...) ]
[ ORDER BY { sort_expression [DESC | ASC]}, ... ]
[ FOR UPDATE [OF { column_expression, ...}] ]
```

## Select Clause

Follow Select with a list of column expressions you want to retrieve or an asterisk (\*) to retrieve all fields.

```
SELECT [DISTINCT] {* | column_expression, [[AS]
column_alias]. . .}
```

*column\_expression* can be simply a field name (for example, LAST\_NAME). More complex expressions may include mathematical operations or string manipulation (for example, SALARY \* 1.05). See [“SQL Expressions” on page 400](#) for details.

*column\_alias* can be used to give the column a more descriptive name. For example, to assign the alias DEPARTMENT to the column DEP:

```
SELECT dep AS department FROM emp
```

Separate multiple column expressions with commas (for example, LAST\_NAME, FIRST\_NAME, HIRE\_DATE).

Field names can be prefixed with the table name or alias. For example, EMP.LAST\_NAME or E.LAST\_NAME, where E is the alias for the table EMP.

The Distinct operator can precede the first column expression. This operator eliminates duplicate rows from the result of a query. For example:

```
SELECT DISTINCT dep FROM emp
```

## Aggregate Functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of records. An aggregate can be used with a field name (for example, AVG(SALARY)) or in combination with a more complex column expression (for example, AVG(SALARY \* 1.07)). The column expression can be preceded by the Distinct operator. The Distinct

operator eliminates duplicate values from an aggregate expression. For example:

```
COUNT (DISTINCT last_name)
```

In this example, only distinct last name values are counted.

[Table A-1](#) lists valid aggregates.

---

**Table A-1. Aggregate Functions**

---

Aggregate	Returns
SUM	The total of the values in a numeric field expression. For example, SUM(SALARY) returns the sum of all salary field values.
AVG	The average of the values in a numeric field expression. For example, AVG(SALARY) returns the average of all salary field values.
COUNT	The number of values in any field expression. For example, COUNT(NAME) returns the number of name values. When using COUNT with a field name, COUNT returns the number of non-null field values. A special example is COUNT(*), which returns the number of records in the set, including records with null values.
MAX	The maximum value in any field expression. For example, MAX(SALARY) returns the maximum salary field value.
MIN	The minimum value in any field expression. For example, MIN(SALARY) returns the minimum salary field value.

---

## From Clause

The From clause indicates the tables that will be used in the Select statement. The format of the From clause is:

```
FROM table_names [table_alias]
```

*table\_names* can be one or more simple table names in the current working directory or complete pathnames.

*table\_alias* is a name used to refer to this table in the rest of the Select statement. Database field names may be prefixed by the table alias. Given the table specification

```
FROM emp E
```

you may refer to the LAST\_NAME field as E.LAST\_NAME. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

If you are joining more than one table, you can use LEFT OUTER JOIN, which includes nonmatching rows in the first table you name. For example:

```
SELECT * FROM T1 LEFT OUTER JOIN T2 on T1.key = T2.key
```

## Where Clause

The Where clause specifies the conditions that records must meet to be retrieved. The Where clause contains conditions in the form:

```
WHERE expr1 rel_operator expr2
```

*expr1* and *expr2* may be field names, constant values, or expressions.

*rel\_operator* is the relational operator that links the two expressions. See [“SQL Expressions” on page 400](#) for details.

For example, the following Select statement retrieves the names of employees that make at least \$20,000.

```
SELECT last_name,first_name FROM emp WHERE salary >= 20000
```

## Group By Clause

The Group By clause specifies the names of one or more fields by which the returned values should be grouped. This clause is used to return a set of aggregate values. It has the following form:

```
GROUP BY column_expressions
```

*column\_expressions* must match the column expression used in the Select clause. A column expression can be one or more field names of the database table, separated by a comma (,) or one or more expressions, separated by a comma (,). See [“SQL Expressions” on page 400](#) for details.

The following example sums the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

## Having Clause

The Having clause enables you to specify conditions for groups of records (for example, display only the departments that have salaries totaling more than \$200,000). This clause is valid only if you have already defined a Group By clause. It has the following form:

```
HAVING expr1 rel_operator expr2
```

*expr1* and *expr2* can be field names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause.

*rel\_operator* is the relational operator that links the two expressions. See [“SQL Expressions” on page 400](#) for details.

The following example returns only the departments whose sums of salaries are greater than \$200,000:

```
SELECT dept_id, sum(salary) FROM emp
GROUP BY dept_id HAVING sum(salary) > 200000
```

## Union Operator

The Union operator combines the results of two Select statements into a single result. The single result is all of the returned records from both Select statements. By default, duplicate records are not returned. To return duplicate records, use the All keyword (UNION ALL). The form is:

```
SELECT statement
UNION [ALL]
SELECT statement
```

When using the Union operator, the select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order. For example:

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

This example has the same number of column expressions, and each column expression, in order, has the same data type.

The following example is *not* valid because the data types of the column expressions are different (SALARY from EMP has a different data type than LAST\_NAME from RAISES). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

## Order By Clause

The Order By clause indicates how the records are to be sorted. The form is:

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

*sort\_expression* can be field names, expressions, or the positional number of the column expression to use.

The default is to perform an ascending (ASC) sort.

For example, to sort by LAST\_NAME then by FIRST\_NAME you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp  
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp  
ORDER BY 2,3
```

In the second example, LAST\_NAME is the second column expression following Select, so Order By 2 sorts by LAST\_NAME.

## For Update Clause

The For Update clause locks the records of the database table selected by the Select statement. The form is:

```
FOR UPDATE OF column_expressions
```

*column\_expressions* is a list of field names in the database table that you intend to update, separated by a comma (,).

The following example returns all records in the employee database that have a SALARY field value of more than \$20,000. When each record is fetched, it is locked. If the record is updated

or deleted, the lock is held until you commit the change. Otherwise, the lock is released when you fetch the next record.

```
SELECT * FROM emp WHERE salary > 20000
FOR UPDATE OF last_name, first_name, salary
```

## SQL Expressions

Expressions are used in the Where clauses, Having clauses, and Order By clauses of SQL Select statements.

Expressions enable you to use mathematical operations as well as character string and date manipulation operators to form complex database queries.

The most common expression is a simple field name. You can combine a field name with other expression elements.

Valid expression elements are as follows:

- Field names
- Constants
- Exponential notation
- Numeric operators
- Character operators
- Date operators
- Relational operators
- Logical operators
- Functions

### ***Constants***

Constants are values that do not change. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant.

You must enclose character constants in pairs of single (') or double quotation marks ("). To include a single quotation mark in a character constant enclosed by single quotation marks, use two single quotation marks together (for example, 'Don''t'). Similarly, if the constant is enclosed by double quotation marks, use two double quotation marks to include one.



You must enclose date and time constants in braces ({}), for example, {01/30/89} and {12:35:10}. The form for date constants is MM/DD/YY or MM/DD/YYYY. The form for time constants is HH:MM:SS.

The logical constants are .T. and 1 for True and .F. and 0 for False. For portability, use 1 and 0.

## ***Exponential Notation***

You may include exponential notation. For example:

```
SELECT col1, 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * col2
```

## ***Numeric Operators***

You may include the following operators in numeric expressions:

<b>Operator</b>	<b>Meaning</b>
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
^	Exponentiation

The following table shows examples of numeric expressions. For these examples, assume SALARY is 20000.

<b>Example</b>	<b>Resulting value</b>
salary + 10000	30000
salary * 1.1	22000
2 ** 3	8

You can precede numeric expressions with a unary plus (+) or minus (-). For example, -(salary \* 1.1) is -22000.

## Character Operators

Character expressions may include the following operators:

Operator	Meaning
+	Concatenation keeping trailing blanks.
-	Concatenation moving trailing blanks to the end.

The following chart shows examples of character expressions. In the examples, LAST\_NAME is 'JONES ' and FIRST\_NAME is 'ROBERT '.

Example	Resulting value
first_name + last_name	'ROBERT JONES '
first_name - last_name	'ROBERTJONES '

NOTE: Some flat-file drivers return character data with trailing blanks as shown in the table; however, you cannot rely on the driver to return blanks. Therefore, if you want an expression that works with drivers that do and do not return trailing blanks, use the TRIM function before concatenating strings to make the expression portable. For example:

```
TRIM(first_name) + ' ' + TRIM(last_name)
```

## Date Operators

You may include the following operators in date expressions:

Operator	Meaning
+	Add a number of days to a date to produce a new date.
-	The number of days between two dates, or subtract a number of days from a date to produce a new date.

The following chart shows examples of date expressions. In these examples, hire\_date is {01/30/1990}.

Example	Resulting value
hire_date + 5	{02/04/1990}
hire_date - {01/01/1990}	29
hire_date - 10	{01/20/1990}

## ***Relational Operators***

The relational operators separating the two expressions may be any one of those listed in [Table A-2](#).

---

***Table A-2. Relational Operators***

---

Operator	Meaning
=	Equal.
<>	Not Equal.
>	Greater Than.
>=	Greater Than or Equal.
<	Less Than.
<=	Less Than or Equal.
Like	Matching a pattern.
Not Like	Not matching a pattern.
Is Null	Equal to Null.
Is Not Null	Not Equal to Null.
Between	Range of values between a lower and upper bound.
In	A member of a set of specified values or a member of a subquery.
Exists	True if a subquery returned at least one record.

**Table A-2. Relational Operators** (cont.)

Operator	Meaning
Any	Compares a value to each value returned by a subquery. Any must be prefaced by =, <>, >, >=, <, or <=.  =Any is equivalent to In.
All	Compares a value to each value returned by a subquery. All must be prefaced by =, <>, >, >=, <, or <=.

The following list shows some examples of relational operators:

```
salary <= 40000
dept = 'D101'
hire_date > {01/30/1989}
salary + commission >= 50000
last_name LIKE 'Jo%'
salary IS NULL
salary BETWEEN 10000 AND 20000
WHERE salary = ANY (SELECT salary FROM emp WHERE dept = 'D101')
WHERE salary > ALL (SELECT salary FROM emp WHERE dept = 'D101')
```

## Logical Operators

Two or more conditions may be combined to form more complex criteria. When two or more conditions are present, they must be related by AND or OR. For example:

```
salary = 40000 AND exempt = 1
```

The logical NOT operator is used to reverse the meaning. For example:

```
NOT (salary = 40000 AND exempt = 1)
```

## Operator Precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. [Table A-3](#) shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.

---

**Table A-3. Operator Precedence**

---

Precedence	Operator
1	Unary -, Unary +
2	**
3	*, /
4	+, -
5	=, <>, <, <=, >, >=, Like, Not Like, Is Null, Is Not Null, Between, In, Exists, Any, All
6	Not
7	AND
8	OR

---

The following example shows the importance of precedence:

```
WHERE salary > 40000 OR
hire_date > {01/30/1989} AND
dept = 'D101'
```

Because AND is evaluated first, this query retrieves employees in department D101 hired after January 30, 1989, as well as every employee making more than \$40,000, no matter what department or hire date.

To force the clause to be evaluated in a different order, use parentheses to enclose the conditions to be evaluated first. For example:

```
WHERE (salary > 40000 OR hire_date > {01/30/1989})
AND dept = 'D101'
```

retrieves employees in department D101 that either make more than \$40,000 or were hired after January 30, 1989.

## Functions

The flat-file drivers support a number of functions that you may use in expressions. In [Table A-4](#) through [Table A-6](#) on [page 410](#), the functions are grouped according to the type of result they return.

---

**Table A-4. Functions that Return Character Strings**

---

Function	Description
CHR	Converts an ASCII code into a one-character string. CHR(67) returns C.
RTRIM	Removes trailing blanks from a string. RTRIM('ABC ') returns ABC.
TRIM	Removes trailing blanks from a string. TRIM('ABC ') returns ABC.
LTRIM	Removes leading blanks from a string. LTRIM(' ABC') returns ABC.
UPPER	Changes each letter of a string to uppercase. UPPER('Allen') returns ALLEN.
LOWER	Changes each letter of a string to lowercase. LOWER('Allen') returns allen.
LEFT	Returns leftmost characters of a string. LEFT('Mattson', 3) returns Mat.

**Table A-4. Functions that Return Character Strings (cont.)**

Function	Description
RIGHT	Returns rightmost characters of a string. RIGHT('Mattson',4) returns tson.
SUBSTR	Returns a substring of a string. Parameters are the string, the first character to extract, and the number of characters to extract (optional). SUBSTR('Conrad',2,3) returns onr. SUBSTR('Conrad',2) returns onrad.
SPACE	Generates a string of blanks. SPACE(5) returns '      '.
DTOC	Converts a date to a character string. An optional second parameter determines the format of the result: 0 (the default) returns MM/DD/YY 1 returns DD/MM/YY 2 returns YY/MM/DD 10 returns MM/DD/YYYY 11 returns DD/MM/YYYY 12 returns YYYY/MM/DD An optional third parameter specifies the date separator character. If not specified, a slash (/) is used. DTOC({01/30/1997}) returns 01/30/97 DTOC({01/30/1997}, 0) returns 01/30/97 DTOC({01/30/1997}, 1) returns 30/01/97 DTOC({01/30/1997}, 2, '-') returns 97-01-30
DTOS	Converts a date to a character string using the format YYYYMMDD. DTOS({01/23/1990}) returns 19900123.
IIF	Returns one of two values. Parameters are a logical expression, the true value, and the false value. If the logical expression evaluates to True, the function returns the true value. Otherwise, it returns the false value. IIF(salary>20000, 'BIG', 'SMALL') returns BIG if SALARY is greater than 20000. If not, it returns SMALL.

**Table A-4. Functions that Return Character Strings** (cont.)

Function	Description
STR	<p>Converts a number to a character string. Parameters are the number, the total number of output characters (including the decimal point), and optionally the number of digits to the right of the decimal point.</p> <p>STR(12.34567, 4) returns 12</p> <p>STR(12.34567, 4, 1) returns 12.3</p> <p>STR(12.34567, 6, 3) returns 12.346</p>
STRVAL	<p>Converts a value of any type to a character string.</p> <p>STRVAL('Woltman') returns Woltman</p> <p>STRVAL({12/25/1953}) returns 12/25/1953</p> <p>STRVAL (5 * 3) returns 15</p> <p>STRVAL (4 = 5) returns 'False'</p>
TIME	<p>Returns the time of day as a string.</p> <p>At 9:49 PM, TIME() returns 21:49:00</p>
TTOC	<p>NOTE: This function is applicable only for those flat-file drivers that support SQL_TIMESTAMP, the Btrieve, Excel 4, Excel 5, FoxPro 3.0, and Paradox 5 drivers.</p> <p>Converts a timestamp to a character string. An optional second parameter determines the format of the result:</p> <p>When set to 0 or none (the default), MM/DD/YY HH:MM:SS AM is returned.</p> <p>When set to 1, YYYYMMDDHHMMSS is returned, which is a suitable format for indexing.</p> <p>TTOC({1992-04-02 03:27:41}) returns 04/02/92 03:27:41 AM.</p> <p>TTOC({1992-04-02 03:27:41, 1}) returns 19920402032741</p>
USERNAME	<p>For Btrieve, the logon ID specified at connect time is returned. For Paradox and Paradox 5 drivers, the user name specified during configuration is returned. For all other flat file drivers, an empty string is returned.</p>



**Table A-5. Functions that Return Numbers**

<b>Function</b>	<b>Description</b>
MOD	Divides two numbers and returns the remainder of the division. MOD(10, 3) returns 1
LEN	Returns the length of a string. LEN('ABC') returns 3
MONTH	Returns the month part of a date. MONTH({01/30/1989}) returns 1
DAY	Returns the day part of a date. DAY({01/30/1989}) returns 30
YEAR	Returns the year part of a date. YEAR({01/30/1989}) returns 1989
MAX	Returns the larger of two numbers. MAX(66, 89) returns 89
DAYOFWEEK	Returns the day of week (1-7) of a date expression. DAYOFWEEK({05/01/1995}) returns 5.
MIN	Returns the smaller of two numbers. MIN(66, 89) returns 66
POW	Raises a number to a power. POW(7, 2) returns 49
INT	Returns the integer part of a number. INT(6.4321) returns 6
ROUND	Rounds a number. ROUND(123.456, 0) returns 123 ROUND(123.456, 2) returns 123.46 ROUND(123.456, -2) returns 100
NUMVAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned. NUMVAL('123') returns the number 123
VAL	Converts a character string to a number. If the character string is not a valid number, a zero is returned. VAL('123') returns the number 123

**Table A-6. Functions that Return Dates**

Function	Description
DATE	Returns today's date. If today is 12/25/1999, DATE() returns {12/25/1999}
TODAY	Returns today's date. If today is 12/25/1999, TODAY() returns {12/25/1999}
DATEVAL	Converts a character string to a date. DATEVAL('01/30/1989') returns {01/30/1989}
CTOD	Converts a character string to a date. An optional second parameter specifies the format of the character string: 0 (the default) returns MM/DD/YY, 1 returns DD/MM/YY, and 2 returns YY/MM/DD. CTOD('01/30/1989') returns {01/30/1989} CTOD('01/30/1989',1) returns {30/01/1989}

The following examples use some of the number and date functions.

Retrieve all employees that have been with the company at least 90 days:

```
SELECT first_name, last_name FROM emp
WHERE DATE() - hire_date >= 90
```

Retrieve all employees hired in January of this year or last year:

```
SELECT first_name, last_name FROM emp
WHERE MONTH(hire_date) = 1
AND (YEAR(hire_date) = YEAR(DATE())
OR YEAR(hire_date) = YEAR(DATE()) - 1)
```

---

# Create and Drop Table Statements

The flat-file drivers support SQL statements to create and delete database files. The Create Table statement is used to create files and the Drop Table statement is used to delete files.

## Create Table

The form of the Create Table statement is:

```
CREATE TABLE table_name (col_definition[,col_definition,...])
```

*table\_name* can be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table 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 .odbc.ini. 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, followed by an optional column constraint definition. Values for column names are database specific. The data type specifies a column's data type.

The only column constraint definition currently supported by some flat-file drivers is "not null." Not all flat-file tables support "not null" columns. In the cases where "not null" is not supported, this restriction is ignored and the driver returns a warning if "not null" is specified for a column. The "not null" column constraint definition is allowed in the driver so that you can write a database-independent application (and not be concerned about the driver raising an error on a Create Table statement with a "not null" restriction).

A sample Create Table statement to create an employee database table is:

```
CREATE TABLE emp (last_name CHAR(20) NOT NULL,  
    first_name CHAR(12) NOT NULL,  
    salary NUMERIC (10,2) NOT NULL,  
    hire_date DATE NOT NULL)
```

## Drop Table

The form of the Drop Table statement is:

```
DROP TABLE table_name
```

*table\_name* may be a simple table name (EMP) or a full pathname. A simple table name is preferred for portability to other SQL data sources. If it is a simple table name, the file is dropped from the directory you specified as the database directory in the connection string. If you did not specify a database directory in the connection string, the file is deleted from the directory you specified as the database directory in .odbc.ini. If you did not specify a database directory in either of these places, the file is dropped from the current working directory at the time you connected to the driver.

A sample Drop Table statement to delete the employee database table is:

```
DROP TABLE emp
```

---

# Insert Statement

The SQL Insert statement is used to add new records to a database table. With it, you can specify either of the following options:

- A list of values to be inserted as a new record
- A Select statement that copies data from another table to be inserted as a set of new records

The form of the Insert statement is:

```
INSERT INTO table_name [(col_name, ...)]  
{VALUES (expr, ...) | select_statement}
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

*col\_name* is an optional list of column names giving the name and order of the columns whose values are specified in the Values clause. If you omit *col\_name*, the value expressions (*expr*) must provide values for all columns defined in the file and must be in the same order that the columns are defined for the file.

*expr* is the list of expressions giving the values for the columns of the new record. Usually, the expressions are constant values for the columns. Character string values must be enclosed in single or double quotation marks, date values must be enclosed in braces {}, and logical values that are letters must be enclosed in periods (for example, .T. or .F.).

An example of an Insert statement that uses a list of expressions is:

```
INSERT INTO emp (last_name, first_name, emp_id, salary, hire_date)  
VALUES ('Smith', 'John', 'E22345', 27500, {4/6/1999})
```

Each Insert statement adds one record to the database table. In this case a record has been added to the employee database table, EMP. Values are specified for five columns. The remaining columns in the table are assigned a blank value, meaning Null.

*select\_statement* is a query that returns values for each *col\_name* value specified in the column name list. Using a Select statement instead of a list of value expressions lets you select a set of rows from one table and insert it into another table using a single Insert statement.

An example of an Insert statement that uses a Select statement is:

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept, salary)
SELECT first_name, last_name, emp_id, dept, salary from emp
WHERE dept = 'D050'
```

In this type of Insert statement, the number of columns to be inserted must match the number of columns in the Select statement. The list of columns to be inserted must correspond to the columns in the Select statement just as it would to a list of value expressions in the other type of Insert statement. That is, the first column inserted corresponds to the first column selected; the second inserted to the second, etc.

The size and data type of these corresponding columns must be compatible. Each column in the Select list should have a data type that the ODBC driver accepts on a regular Insert/Update of the corresponding column in the Insert list. Values are truncated when the size of the value in the Select list column is greater than the size of the corresponding Insert list column.

The *select\_statement* is evaluated before any values are inserted. This query cannot be made on the table into which values are inserted.

---

# Update Statement

The SQL Update statement is used to change records in a database file. The form of the Update statement supported for flat-file drivers is:

```
UPDATE table_name SET col_name = expr, ...  
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

*col\_name* is the name of a column whose value is to be changed. Several columns can be changed in one statement.

*expr* is the new value for the column. The expression can be a constant value or a subquery. Character string values must be enclosed with single or double quotation marks, date values must be enclosed by braces {}, and logical values that are letters must be enclosed by periods (for example, .T. or .F.). Subqueries must be enclosed in parentheses.

The Where clause (any valid clause described in [“Select Statement” on page 393](#)) determines which records are to be updated.

The Where Current Of *cursor\_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor\_name* is positioned to be updated. This is called a "positional update." You must first execute a Select...For Update statement with a named cursor and fetch the row to be updated.

An example of an Update statement on the employee table is:

```
UPDATE emp SET salary=32000, exempt=1  
WHERE emp_id = 'E10001'
```

The Update statement changes every record that meets the conditions in the Where clause. In this case the salary and exempt status are changed for all employees having the employee ID E10001. Because employee IDs are unique in the employee table, only one record is updated.

An example using a subquery is:

```
UPDATE emp SET salary = (SELECT avg(salary) from emp)
WHERE emp_id = 'E10001'
```

In this case, the salary is changed to the average salary in the company for the employee having employee ID E10001.

---

## Delete Statement

The SQL Delete statement is used to delete records from a database table. The form of the Delete statement supported for flat-file drivers is:

```
DELETE FROM table_name
[ WHERE { conditions | CURRENT OF cursor_name } ]
```

*table\_name* may be a simple table name or a full pathname. A simple table name is preferred for portability to other SQL data sources.

The Where clause (any valid clause described in [“Select Statement” on page 393](#)) determines which records are to be deleted. If you include only the keyword Where, all records in the table are deleted but the file is left intact.

The Where Current Of *cursor\_name* clause can be used only by developers coding directly to the ODBC API. It causes the row at which *cursor\_name* is positioned to be deleted. This is called a “positional delete.” You must first execute a Select...For Update statement with a named cursor and fetch the row to be deleted.



An example of a Delete statement on the employee table is:

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

Each Delete statement removes every record that meets the conditions in the Where clause. In this case every record having the employee ID E10001 is deleted. Because employee IDs are unique in the employee table, at most, one record is deleted.

---

## Reserved Keywords

The following words are reserved for use in SQL statements. If they are used for file or column names in a database that you use, you must enclose them in quotation marks in any SQL statement where they appear as file or column names.

- |            |          |           |         |
|------------|----------|-----------|---------|
| ■ ALL      | ■ FROM   | ■ LIKE    | ■ OR    |
| ■ AND      | ■ FULL   | ■ NATURAL | ■ ORDER |
| ■ BETWEEN  | ■ GROUP  | ■ NOT     | ■ RIGHT |
| ■ COMPUTE  | ■ HAVING | ■ NULL    | ■ UNION |
| ■ CROSS    | ■ INNER  | ■ ON      | ■ WHERE |
| ■ DISTINCT | ■ INTO   | ■ OPTIONS |         |
| ■ FOR      | ■ LEFT   | ■ OR      |         |



# B Using Indexes

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

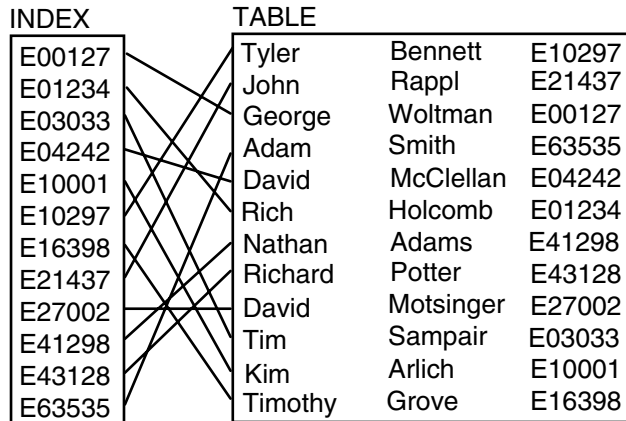
For information regarding how to create and drop indexes, see the appropriate database driver chapter for flat-file drivers or your database system documentation for relational drivers.

---

## Introduction

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

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



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

```
WHERE emp_id = 'E10001'
```

Without an index, the driver must search the entire database table to find those records having an employee ID of E10001. By using an index on the EMP\_ID field, however, the driver can quickly find those records.

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

---

## Improving Record Selection Performance

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

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

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

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

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

---

## Indexing Multiple Fields

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

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

For this condition, the optimal index field expression is `LAST_NAME, FIRST_NAME`. This creates a concatenated index.

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

```
last_name = 'Smith'
```

Consider the following Where clause:

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

If your index fields include all the conditions of the Where clause in that order, the driver can use the entire index. If, however, your index is on two nonconsecutive fields, say, `LAST_NAME` and `FIRST_NAME`, the driver can use only the `LAST_NAME` field of the index.

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

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

In this case, the driver selects the index on the `LAST_NAME` field.

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

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

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

---

## Deciding Which Indexes to Create

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

- Insert, update, and delete records
- Retrieve records

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

If you most often retrieve records, you must look further to define the criteria for retrieving records and create indexes to improve the performance of these retrievals. Assume you have an employee database table and you will retrieve records based on employee name, department, or hire date. You would create three indexes—one on the DEPT field, one on the HIRE\_DATE field, and one on the LAST\_NAME field. Or perhaps, for the retrievals based on the name field, you would want an index that concatenates the LAST\_NAME and the FIRST\_NAME fields (see [“Indexing Multiple Fields” on page 421](#) for details).

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

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

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

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

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

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



---

# Improving Join Performance

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

Assume you have the following Select statement:

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

In this example, the DEPT and EMP database tables are being joined using the department ID field. When the driver executes a query that contains a join, it processes the tables from left to right and uses an index on the second table's join field (the DEPT field of the EMP table).

To improve join performance, you need an index on the join field of the second table in the From clause. If there is a third table in the From clause, the driver also uses an index on the field in the third table that joins it to any previous table. For example:

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

In this case, you should have an index on the EMP.DEPT field and the ADDR.LOC field.



# C ODBC API and Scalar Functions

This appendix lists the ODBC API functions that the Connect ODBC drivers support and the scalar functions, which you use in SQL statements.

---

## API Functions

All Connect ODBC drivers are ODBC Level 1–compliant—they support all ODBC Core and Level 1 functions. They also support a limited set of Level 2 functions. The drivers support the functions listed in [Table C-1 on page 428](#) and [Table C-2 on page 429](#). Any additions to these supported functions or differences in the support of specific functions are listed in the "ODBC Conformance Level" section in the individual driver chapters.

**Table C-1. Function Conformance for 2.x ODBC Applications**

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

---

**Table C-2. Function Conformance for 3.x ODBC Applications**


---

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

---

---

## Scalar Functions

This section lists the scalar functions that ODBC supports. Your database system may not support all of these functions. See the documentation for your database system to find out which functions are supported.

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

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

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

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

## String Functions

[Table C-3 on page 431](#) lists the string functions that ODBC supports.

The string functions listed can take the following arguments:

- *string\_exp* can be the name of a column, a string literal, or the result of another scalar function, where the underlying data type is SQL\_CHAR, SQL\_VARCHAR, or SQL\_LONGVARCHAR.
- *start*, *length*, and *count* can be the result of another scalar function or a literal numeric value, where the underlying data type is SQL\_TINYINT, SQL\_SMALLINT, or SQL\_INTEGER.

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

Character string literals must be surrounded in single quotation marks.

**Table C-3. Scalar String Functions**

Function	Returns
ASCII( <i>string_exp</i> )	ASCII code value of the leftmost character of <i>string_exp</i> as an integer.
BIT_LENGTH( <i>string_exp</i> ) [ODBC 3.0 only]	The length in bits of the string expression.
CHAR( <i>code</i> )	The character with the ASCII code value specified by <i>code</i> . <i>code</i> should be between 0 and 255; otherwise, the return value is data-source dependent.
CHAR_LENGTH( <i>string_exp</i> ) [ODBC 3.0 only]	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHARACTER_LENGTH function.)
CHARACTER_LENGTH( <i>string_exp</i> ) [ODBC 3.0 only]	The length in characters of the string expression, if the string expression is of a character data type; otherwise, the length in bytes of the string expression (the smallest integer not less than the number of bits divided by 8). (This function is the same as the CHAR_LENGTH function.)
CONCAT( <i>string_exp1</i> , <i>string_exp2</i> )	The string resulting from concatenating <i>string_exp2</i> and <i>string_exp1</i> . The string is system dependent.
DIFFERENCE( <i>string_exp1</i> , <i>string_exp2</i> )	An integer value that indicates the difference between the values returned by the SOUNDEX function for <i>string_exp1</i> and <i>string_exp2</i> .
INSERT( <i>string_exp1</i> , <i>start</i> , <i>length</i> , <i>string_exp2</i> )	A string where <i>length</i> characters have been deleted from <i>string_exp1</i> beginning at <i>start</i> and where <i>string_exp2</i> has been inserted into <i>string_exp</i> , beginning at <i>start</i> .
LCASE( <i>string_exp</i> )	Uppercase characters in <i>string_exp</i> converted to lowercase.
LEFT( <i>string_exp</i> , <i>count</i> )	The <i>count</i> of characters of <i>string_exp</i> .
LENGTH( <i>string_exp</i> )	The number of characters in <i>string_exp</i> , excluding trailing blanks and the string termination character.

**Table C-3. Scalar String Functions** (cont.)

Function	Returns
LOCATE( <i>string_exp1</i> , <i>string_exp2</i> [, <i>start</i> ])	The starting position of the first occurrence of <i>string_exp1</i> within <i>string_exp2</i> . If <i>start</i> is not specified, the search begins with the first character position in <i>string_exp2</i> . If <i>start</i> is specified, the search begins with the character position indicated by the value of <i>start</i> . The first character position in <i>string_exp2</i> is indicated by the value 1. If <i>string_exp1</i> is not found, 0 is returned.
LTRIM( <i>string_exp</i> )	The characters of <i>string_exp</i> , with leading blanks removed.
OCTET_LENGTH( <i>string_exp</i> ) [ODBC 3.0 only]	The length in bytes of the string expression. The result is the smallest integer not less than the number of bits divided by 8.
POSITION( <i>character_exp</i> IN <i>character_exp</i> ) [ODBC 3.0 only]	The position of the first character expression in the second character expression. The result is an exact numeric with an implementation-defined precision and a scale of 0.
REPEAT( <i>string_exp</i> , <i>count</i> )	A string composed of <i>string_exp</i> repeated <i>count</i> times.
REPLACE( <i>string_exp1</i> , <i>string_exp2</i> , <i>string_exp3</i> )	Replaces all occurrences of <i>string_exp2</i> in <i>string_exp1</i> with <i>string_exp3</i> .
RIGHT( <i>string_exp</i> , <i>count</i> )	The rightmost <i>count</i> of characters in <i>string_exp</i> .
RTRIM( <i>string_exp</i> )	The characters of <i>string_exp</i> with trailing blanks removed.
SOUNDEX( <i>string_exp</i> )	A data-source-dependent string representing the sound of the words in <i>string_exp</i> .
SPACE( <i>count</i> )	A string consisting of <i>count</i> spaces.
SUBSTRING( <i>string_exp</i> , <i>start</i> , <i>length</i> )	A string derived from <i>string_exp</i> beginning at the character position <i>start</i> for <i>length</i> characters.
UCASE( <i>string_exp</i> )	Lowercase characters in <i>string_exp</i> converted to uppercase.



## Numeric Functions

Table C-4 lists the numeric functions that ODBC supports.

The numeric functions listed can take the following arguments:

- *numeric\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_NUMERIC, SQL\_DECIMAL, SQL\_TINYINT, SQL\_SMALLINT, SQL\_INTEGER, SQL\_BIGINT, SQL\_FLOAT, SQL\_REAL, or SQL\_DOUBLE.
- *float\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_FLOAT.
- *integer\_exp* can be a column name, a numeric literal, or the result of another scalar function, where the underlying data type is SQL\_TINYINT, SQL\_SMALLINT, SQL\_INTEGER, or SQL\_BIGINT.

---

**Table C-4. Scalar Numeric Functions**

---

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

**Table C-4. Scalar Numeric Functions** (cont.)

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

## Date and Time Functions

Table C-5 lists the date and time functions that ODBC supports.

The date and time functions listed can take the following arguments:

- *date\_exp* can be a column name, a date or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_DATE, or SQL\_TIMESTAMP.
- *time\_exp* can be a column name, a timestamp or timestamp literal, or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_TIME, or SQL\_TIMESTAMP.
- *timestamp\_exp* can be a column name; a time, date, or timestamp literal; or the result of another scalar function, where the underlying data type can be represented as SQL\_CHAR, SQL\_VARCHAR, SQL\_TIME, SQL\_DATE, or SQL\_TIMESTAMP.

---

**Table C-5. Scalar Time and Date Functions**

---

Function	Returns
CURRENT_DATE() <i>[ODBC 3.0 only]</i>	Current date.
CURRENT_TIME[( <i>time-precision</i> )] <i>[ODBC 3.0 only]</i>	Current local time. The <i>time-precision</i> argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[( <i>timestamp-precision</i> )] <i>[ODBC 3.0 only]</i>	Current local date and local time as a timestamp value. The <i>timestamp-precision</i> argument determines the seconds precision of the returned timestamp.
CURDATE()	Current date as a date value.
CURTIME()	Current local time as a time value.

**Table C-5. Scalar Time and Date Functions** (cont.)

Function	Returns
DAYNAME( <i>date_exp</i> )	Character string containing a data-source-specific name of the day for the day portion of <i>date_exp</i> .
DAYOFMONTH( <i>date_exp</i> )	Day of the month in <i>date_exp</i> as an integer value (1–31).
DAYOFWEEK( <i>date_exp</i> )	Day of the week in <i>date_exp</i> as an integer value (1–7).
DAYOFYEAR( <i>date_exp</i> )	Day of the year in <i>date_exp</i> as an integer value (1–366).
HOUR( <i>time_exp</i> )	Hour in <i>time_exp</i> as an integer value (0–23).
MINUTE( <i>time_exp</i> )	Minute in <i>time_exp</i> as an integer value (0–59).
MONTH( <i>date_exp</i> )	Month in <i>date_exp</i> as an integer value (1–12).
MONTHNAME( <i>date_exp</i> )	Character string containing the data source-specific name of the month.
NOW()	Current date and time as a timestamp value.
QUARTER( <i>date_exp</i> )	Quarter in <i>date_exp</i> as an integer value (1–4).
SECOND( <i>time_exp</i> )	Second in <i>date_exp</i> as an integer value (0–59).
TIMESTAMPADD( <i>interval</i> , <i>integer_exp</i> , <i>time_exp</i> )	Timestamp calculated by adding <i>integer_exp</i> intervals of type <i>interval</i> to <i>time_exp</i> . <i>interval</i> can be SQL_TSI_FRAC_SECOND SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR Fractional seconds are expressed in billionths of a second.

**Table C-5. Scalar Time and Date Functions** (cont.)

Function	Returns
TIMESTAMPDIFF( <i>interval</i> , <i>time_exp1</i> , <i>time_exp2</i> )	Integer number of intervals of type <i>interval</i> by which <i>time_exp2</i> is greater than <i>time_exp1</i> . <i>interval</i> has the same values as TIMESTAMPADD. Fractional seconds are expressed in billionths of a second.
WEEK( <i>date_exp</i> )	Week of the year in <i>date_exp</i> as an integer value (1–53).
YEAR( <i>date_exp</i> )	Year in <i>date_exp</i> . The range is data-source dependent.

## System Functions

[Table C-6](#) lists the system functions that ODBC supports.

**Table C-6. Scalar System Functions**

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



# D Locking and Isolation Levels

This appendix discusses locking and isolation levels and how their settings can affect the data you retrieve. Different database systems support different locking and isolation levels. See the section "Isolation and Lock Levels Supported" in the appropriate driver chapter.

---

## Locking

Locking is a database operation that restricts a user from accessing a table or record. Locking is used in situations where more than one user might try to use the same table or record at the same time. By locking the table or record, the system ensures that only one user at a time can affect the data.

Locking is critical in multiuser databases, where different users can try to access or modify the same records concurrently. Although such concurrent database activity is desirable, it can create problems. Without locking, for example, if two users try to modify the same record at the same time, they might encounter problems ranging from retrieving bad data to deleting data that the other user needs. If, however, the first user to access a record can lock that record to temporarily prevent other users from modifying it, such problems can be avoided. Locking provides a way to manage concurrent database access while minimizing the various problems it can cause.

---

## Isolation Levels

An isolation level represents a particular locking strategy employed in the database system to improve data consistency. The higher the isolation level, the more complex the locking strategy behind it. The isolation level provided by the database determines whether a transaction will encounter the following behaviors in data consistency:

Dirty reads	User 1 modifies a row. User 2 reads the same row before User 1 commits. User 1 performs a rollback. User 2 has read a row that has never really existed in the database. User 2 may base decisions on false data.
Non-repeatable reads	User 1 reads a row but does not commit. User 2 modifies or deletes the same row and then commits. User 1 rereads the row and finds it has changed (or has been deleted).
Phantom reads	User 1 uses a search condition to read a set of rows but does not commit. User 2 inserts one or more rows that satisfy this search condition, then commits. User 1 rereads the rows using the search condition and discovers rows that were not present before.

Isolation levels represent the database system's ability to prevent these behaviors. The American National Standards Institute (ANSI) defines four isolation levels:

- Read uncommitted (0)
- Read committed (1)
- Repeatable read (2)
- Serializable (3)



In ascending order (0–3), these isolation levels provide an increasing amount of data consistency to the transaction. At the lowest level, all three behaviors can occur. At the highest level, none can occur. The success of each level in preventing these behaviors is due to the locking strategies that they employ, which are as follows:

- |                      |   |
|----------------------|---|
| Read uncommitted (0) | Locks are obtained on modifications to the database and held until end of transaction (EOT). Reading from the database does not involve any locking.  |
| Read committed (1)   | Locks are acquired for reading and modifying the database. Locks are released after reading but locks on modified objects are held until EOT.   |
| Repeatable read (2)  | Locks are obtained for reading and modifying the database. Locks on all modified objects are held until EOT. Locks obtained for reading data are held until EOT. Locks on non-modified access structures (such as indexes and hashing structures) are released after reading. |
| Serializable (3)     | All data read or modified is locked until EOT. All access structures that are modified are locked until EOT. Access structures used by the query are locked until EOT.  |

Table D-1 shows what data consistency behaviors can occur at each isolation level.

---

**Table D-1. Isolation Levels and Data Consistency**

---

Level	Dirty Read	Nonrepeatable Read	Phantom Read
0, Read uncommitted	Yes	Yes	Yes
1, Read committed	No	Yes	Yes
2, Repeatable read	No	No	Yes
3, Serializable	No	No	No

---

Although higher isolation levels provide better data consistency, this consistency can be costly in terms of the concurrency provided to individual users. Concurrency is the ability of multiple users to access and modify data simultaneously. As isolation levels increase, so does the chance that the locking strategy used will create problems in concurrency.

*Put another way:* The higher the isolation level, the more locking involved, and the more time users may spend waiting for data to be freed by another user. Because of this inverse relationship between isolation levels and concurrency, you must consider how people use the database before choosing an isolation level. You must weigh the trade-offs between data consistency and concurrency, and decide which is more important.

---

## Locking Modes and Levels

Different database systems employ various locking modes, but they have two basic ones in common: shared and exclusive. Shared locks can be held on a single object by multiple users. If one user has a shared lock on a record, then a second user can also get a shared lock on that same record; however, the second user cannot get an exclusive lock on that record. Exclusive locks are exclusive to the user that obtains them. If one user has an exclusive lock on a record, then a second user cannot get either type of lock on the same record.

Performance and concurrency can also be affected by the locking level used in the database system. The locking level determines the size of an object that is locked in a database. For example, many database systems let you lock an entire table, as well as individual records. An intermediate level of locking, page-level locking, is also common. A page contains one or more records and is typically the amount of data read from the disk in a single disk access. The major disadvantage of page-level locking is that if one user locks a record, a second user may not be able to lock other records because they are stored on the same page as the locked record.



# E Threading

The ODBC specification mandates that all drivers must be thread-safe; that is, drivers must not fail when database requests are made on separate threads. It is a common misperception that issuing requests on separate threads will always result in improved throughput. Because of network transport and database server limitations, some drivers may serialize threaded requests to the server to ensure thread safety.

The ODBC 3.0 specification does not provide a method to find out how a driver will service threaded requests although this information is quite useful to an application. All DataDirect drivers provide this information to the user via the SQLGetInfo information type 1028.

The result of calling SQLGetInfo with 1028 is a SQL\_USMALLINT flag which denotes the session's thread model. A return value of 0 denotes that the session is fully thread enabled and that all requests will fully utilize the threaded model. A return value of 1 denotes that the session is restricted at the connection level. Sessions of this type are fully thread-enabled when simultaneous threaded requests are made with statement handles that do not share the same connection handle. In this model, if multiple requests are made from the same connection, then the first request received by the driver is processed immediately and all subsequent requests are serialized. A return value of 2 denotes that the session is thread-impaired and all requests are serialized by the driver.

Consider the following code fragment:

```
rc = SQLGetInfo (hdbc, 1028, &ThreadModel, NULL, NULL);

If (rc == SQL_SUCCESS) {
    // driver is a MERANT driver which can report
    // threading information

    if (ThreadModel == 0)
        // driver is unconditionally thread enabled
        // application can take advantage of threading

    else if (ThreadModel == 1)
        // driver is thread enabled when thread requests are
        // from different connections
        // some applications can take advantage of threading

    else if (ThreadModel == 2)
        // driver is thread impaired
        // application should only use threads if it reduces
        // program complexity

}
else
    // driver is only guaranteed to be thread-safe
    // use threading at your own risk
```

[Table E-1](#) summarizes the threading information available at this time for DataDirect drivers. Always consult the README file for the most up-to-date information as the threading information is subject to change with new database transport and server revisions.

---

**Table E-1. Threading Information**

---

<b>Driver</b>	<b>Fully Threaded</b>	<b>Thread Per Connect</b>	<b>Thread Impaired</b>
Btrieve	✓		
dBASE	✓		
DB2		✓	
Excel Workbook	✓		
Informix		✓	
Informix Dynamic Server		✓	
MySQL			✓
Oracle		✓	
Paradox	✓		
PROGRESS			✓
SQLBase		✓	
SQL Server		✓	
Sybase ASE		✓	
Text	✓		
XML			✓

---





# F Performance Design of ODBC Applications

This appendix provides information about performance issues and guidelines for developing performance-optimized, ODBC applications for ODBC/OLE DB Adapter and ODBC drivers.

---

## Optimizing Performance

Developing performance-oriented ODBC applications is not easy. Microsoft's *ODBC Programmer's Reference* does not provide information about system performance. In addition, ODBC drivers and the ODBC driver manager do not return warnings when applications run inefficiently.

The following sections contain guidelines compiled by examining the ODBC implementations of numerous shipping ODBC applications.

[Table F-1](#) summarizes some common ODBC system performance problems and possible solutions.

---

***Table F-1. Common ODBC System Performance Problems and Solutions***

---

<b>Problem</b>	<b>Solution</b>
Network communication is slow	Reduce network traffic
The process of evaluating complex SQL queries on the database server is slow and might reduce concurrency	Simplify queries

---

**Table F-1. Common ODBC System Performance Problems and Solutions** (cont.)

---

<b>Problem</b>	<b>Solution</b>
Excessive calls from the application to the driver decreases performance	Optimize application-to-driver interaction
Disk input/output is slow	Limit disk input/output

---

The guidelines are divided into five sections: [“Catalog Functions”](#) on page 450, [“Retrieving Data”](#) on page 454, [“ODBC Function Selection”](#) on page 459, [“Design Options”](#) on page 462, and [“Updating Data”](#) on page 465.

---

## Catalog Functions

The following ODBC functions are defined to be catalog functions:

- SQLColumns
- SQLColumnPrivileges
- SQLForeignKeys
- SQLGetTypeInfo
- SQLProcedures
- SQLProcedureColumns
- SQLSpecialColumns
- SQLStatistics
- SQLTables

While some drivers implement `SQLGetTypeInfo` as hard-coded information, many drivers must query the server to obtain accurate information about which types are supported (for example, to find dynamic types such as user defined types, and so on). Therefore, `SQLGetTypeInfo` is included in this list of potentially expensive ODBC functions.

## Catalog Functions Are Relatively Slow

Catalog functions are relatively slow compared to other ODBC functions. Applications should cache information returned from catalog functions so that multiple executions are not needed.

While almost no ODBC application can be written without catalog functions, their use should be minimized. To return all result column information *mandated* by the ODBC specification, a driver may have to perform multiple queries, joins, subqueries, and/or unions in order to return the necessary result set for a single call to a catalog function. These particular elements of the SQL language are performance expenses. Frequent use of catalog functions in an application will likely result in poor performance.

Applications should cache information from catalog functions. For example, call `SQLGetTypeInfo` once in the application and cache away the elements of the result set that your application depends on. It is unlikely that any application uses all elements of the result set generated by a catalog function, so the cache of information should not be difficult to maintain.

## Passing Null Arguments

Passing null arguments to catalog functions results in generating time consuming queries. In addition, network traffic potentially increases due to unwanted result set information. Always supply as many non-null arguments to catalog functions as possible.

Because catalog functions are slow, applications should invoke them as efficiently as possible. Many applications pass the fewest non-null arguments necessary for the function to return success.

For example, consider a call to `SQLTables` where the application requests information about table "Customers." Often, this call is coded in a manner similar to the following example:

```
rc = SQLTables (NULL, NULL, NULL, NULL, "Customers", SQL_NTS, NULL);
```

A driver could turn this SQLTables call into SQL similar to:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers' UNION ALL  
SELECT ... FROM SysViews WHERE ViewName = 'Customers' UNION ALL  
SELECT ... FROM SysSynonyms WHERE SynName = 'Customers'  
ORDER BY ...
```

Sometimes, little information is known about the object for which you are requesting information. Any information that the application can send the driver when calling catalog functions can result in improved performance and reliability.

Using the previous example, suppose three "Customers" tables were returned in the result set:

- The first table was owned by the user
- The second table was owned by the sales department
- The third table was a view created by management

It might not be obvious to the user which table to choose. If the application had specified the OwnerName argument for the SQLTables call, only one table would be returned and performance would increase. This is because less network traffic was required to return only one result row and unwanted rows were filtered by the database.

In addition, if the TableType argument can be supplied, then the SQL sent to the server can be optimized from a three query union to a single, Select statement as shown:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers' and Owner = 'Beth'
```

## SQLColumns

Avoid using SQLColumns to determine characteristics about a table. Instead, use a dummy query with SQLDescribeCol.

Consider an application that allows the user to choose the columns that will be selected. Should the application use SQLColumns to return information about the columns to the user or instead prepare a dummy query and call SQLDescribeCol?

### ***Case 1: SQLColumns Method***

```
rc = SQLColumns (... "UnknownTable" ...);
// This call to SQLColumns will generate a query to the
// system
// catalogs... possibly a join which must be prepared,
// executed, and produce a result set
rc = SQLBindCol (...);
rc = SQLExtendedFetch (...);
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained
```

### ***Case 2: SQLDescribeCol Method***

```
// prepare dummy query
rc = SQLPrepare (... "SELECT * from UnknownTable
    WHERE 1 = 0" ...);
// query is never executed on the server - only prepared
rc = SQLNumResultCols (...);
for (irow = 1; irow <= NumColumns; irow++) {
    rc = SQLDescribeCol (...);
    // + optional calls to SQLColAttributes
}
// result column information has now been obtained
// Note we also know the column ordering within the table!
// This information cannot be
// assumed from the SQLColumns example.
```

In both cases a query is sent to the server, but in Case 1 the query must be evaluated and form a result set that must be sent to the client. Clearly, Case 2 is the better performing model.

To somewhat complicate this discussion, let us consider a DBMS server that does not natively support preparing a SQL statement. The performance of Case 1 does not change but Case 2 increases minutely because the dummy query must be evaluated instead of only prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and should execute without accessing table data. For even this type of driver, method 2 out performs method 1.

---

## Retrieving Data

This section provides information about retrieving data with ODBC applications.

### Retrieving Long Data

Retrieving long data (SQL\_LONGVARCHAR and SQL\_LONGVARBINARY data) across the network is very resource intensive and thus slow. Applications should avoid requesting long data unless it is absolutely necessary.

How often do users want to see long data? Most users don't want to see such information. If the user does wish to see these result items, then the application can requery the database specifying only the long columns in the select list. This method allows the average user to retrieve the result set without having to pay a high performance penalty for network traffic.

Although the best method is to exclude long data from the select list, some applications do not formulate the select list before sending the query to the ODBC driver (that is, some applications

simply `select * from <table name> ...`). If the select list contains long data then some drivers *must* retrieve that data at fetch time even if the application does not bind the long data in the result set. If possible, the designer should attempt to implement a method that does not retrieve all columns of the table.

## Reducing the Size of Data Retrieved

Reduce the size of any data being retrieved to some manageable limit by calling `SQLSetStmtOption` with the `SQL_MAX_LENGTH` option. This reduces network traffic and improves performance.

While eliminating `SQL_LONGVARCHAR` and `SQL_LONGVARBINARY` data from the result set is ideal for performance optimization, sometimes, long data must be retrieved. When this is the case, remember that most users do not want to see 100 KB, or more, of text on the screen. What techniques, if any, are available to limit the amount of data retrieved?

Many application developers mistakenly assume that if they call `SQLGetData` with a container of size  $x$  that the ODBC driver only retrieves  $x$  bytes of information from the server. Because `SQLGetData` can be called multiple times for any one column, most drivers optimize their network use by retrieving long data in large chunks and then returning it to the user when requested. For example:

```
char CaseContainer[1000];
...
rc = SQLExecDirect (hstmt, "SELECT CaseHistory FROM Cases
WHERE
    CaseNo = 71164", SQL_NTS);
...
rc = SQLFetch (hstmt);
rc = SQLGetData (hstmt, 1, CaseContainer, (SWORD)
sizeof(CaseContainer), ...);
```

At this point, it is more likely that an ODBC driver retrieves 64 KB of information from the server instead of 1000 bytes. One 64 KB retrieval is less expensive than sixty-four, 1000-byte retrievals in terms of network access. Unfortunately, the application may not call `SQLGetData` again; thus, the first and only retrieval of `CaseHistory` would be slowed by the fact that 64 KB of data had to be sent across the network.

Many ODBC drivers allow limiting the amount of data retrieved across the network by supporting the statement option `SQL_MAX_LENGTH`. This attribute allows the driver to communicate to the database server that only `Z` bytes of data are pertinent to the client. The server responds by sending only the first `Z` bytes of data for *all* result columns. This optimization greatly reduces network traffic and thus improves performance of the client. Our example returned just one row, but consider the case where 100 rows are returned in the result set. The performance improvement is substantial.

## Using Bound Columns

Retrieving data through bound columns (`SQLBindCol`) instead of using `SQLGetData` reduces the ODBC call load and thus improves performance.

Consider the following pseudo-code fragment:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM
Employees
    WHERE HireDate >= ?", SQL_NTS);
do {
rc = SQLFetch (hstmt);
// call SQLGetData 20 times
} while ((rc == SQL_SUCCESS) || (rc ==
SQL_SUCCESS_WITH_INFO));
```



Suppose the query returns 90 result rows. More than 1890 ODBC calls are made (20 calls to `SQLGetData` x 90 result rows + 91 calls to `SQLFetch`).

Consider the same scenario that uses `SQLBindCol` instead of `SQLGetData`:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM Employees
    WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 20 times
do {
rc = SQLFetch (hstmt);
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made is reduced from more than 1890 to about 110 (20 calls to `SQLBindCol` + 91 calls to `SQLFetch`). In addition to reducing the call load, many drivers optimize use of `SQLBindCol` by binding result information directly from the database server into the user's buffer. That is, instead of the driver retrieving information into a container then copying that information to the user's buffer, the driver simply requests the information from the server be placed directly into the user's buffer.

## Using `SQLExtendedFetch` Instead of `SQLFetch`

Use `SQLExtendedFetch` to retrieve data instead of `SQLFetch`. The ODBC call load decreases (resulting in better performance,) and the code is less complex (resulting in more maintainable code).

Most ODBC drivers now support `SQLExtendedFetch` for forward only cursors; yet, most ODBC applications use `SQLFetch` to retrieve data. Again consider the example above using `SQLExtendedFetch` instead of `SQLFetch`:

```
rc = SQLSetStmtOption (hstmt, SQL_ROWSET_SIZE, 100);
// use arrays of 100 elements
rc = SQLExecDirect (hstmt, "SELECT <20 columns> FROM
  Employees WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 1 time specifying row-wise binding
do {
rc = SQLExtendedFetch (hstmt, SQL_FETCH_NEXT, 0, &RowsFetched,
  RowStatus);
} while ((rc == SQL_SUCCESS) || (rc == SQL_SUCCESS_WITH_INFO));
```

The number of ODBC calls made by the application is reduced from 110 in the last example to four (1 `SQLSetStmtOption` + 1 `SQLExecDirect` + 1 `SQLBindCol` + 1 `SQLExtendedFetch`). Note the total savings from an initial call load of more than 1890 ODBC calls in the first presentation of the example to four above. In addition to reducing the call load, many ODBC drivers retrieve data from the server in arrays that further improves the performance by reducing network traffic.

For ODBC drivers that do not support `SQLExtendedFetch`, the application can enable forward-only cursors using the ODBC cursor library (call `SQLSetConnectOption` using `SQL_ODBC_CURSORS/SQL_CUR_USE_IF_NEEDED`). While using the cursor library does not improve performance, it should not be detrimental to application response time when using forward only cursors (no logging is required). Furthermore, using the cursor library when `SQLExtendedFetch` is not supported natively by the driver simplifies the code because the application can always depend on `SQLExtendedFetch` being available. The application need not code two algorithms (one using `SQLExtendedFetch` and one using `SQLFetch`).

---

## ODBC Function Selection

This section provides guidelines for selecting functions for performance optimization.

### Using SQLPrepare/SQLExecute and SQLExecDirect

Do not assume that SQLPrepare/SQLExecute is always as efficient as SQLExecDirect. Use SQLExecDirect for queries that will be executed once and SQLPrepare/SQLExecute for queries that will be executed more than once.

ODBC drivers are optimized based on the perceived use of the functions that are being executed. SQLPrepare/SQLExecute is optimized for multiple executions of a statement that most likely uses parameter markers. SQLExecDirect is optimized for a single execution of a SQL statement. Unfortunately, more than 75% of all ODBC applications use SQLPrepare/SQLExecute *exclusively*.

The pitfall of always coding SQLPrepare/SQLExecute can be understood better by considering an ODBC driver that implements SQLPrepare by creating a stored procedure on the server that contains the prepared statement. Creating a stored procedure has substantial overhead, but the ODBC driver is assuming is that the statement will be *executed* multiple times. While stored procedure creation is relatively expensive, execution is minimal because the query is parsed and optimization paths are stored at create procedure time. Using SQLPrepare/SQLExecute for a statement that will be executed only once with such an ODBC driver will result in unneeded overhead. Furthermore, applications that use SQLPrepare/SQLExecute for large single execution query batches

will almost certainly exhibit poor performance when used with ODBC drivers as previously discussed.

Similar arguments can be used to show applications that always use `SQLExecDirect` cannot perform as well as those that logically use a combination of `SQLPrepare/SQLExecute` and `SQLExecDirect` sequences.

## Using `SQLPrepare` and Multiple `SQLExecute` Calls

Applications that use `SQLPrepare` and multiple `SQLExecute` calls should use `SQLParamOptions` if available. Passing arrays of parameter values reduces the ODBC call load and greatly reduces network traffic.

Consider the following example designed to insert data:

```
rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger (...)
VALUES
    (?, ?, ...) ", SQL_NTS);
// bind parameters
...
do {
// read ledger values into bound parameter buffers
...
rc = SQLExecute (hstmt);      // insert row
} while ! (eof);
```

If there are 100 rows to insert then `SQLExecute` is called 100 times resulting in 100 network requests to the server. Consider, however, an algorithm that uses parameter arrays by calling `SQLParamOptions`:

```

rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger (...)
VALUES
    (?, ?, ...) ", SQL_NTS);
rc = SQLParamOptions (hstmt, (UDWORD) 50, &CurrentRow);
// pass 50 parameters per execute
// bind parameters
...
do {
// read up to 50 ledger values into bound parameter buffers
...
rc = SQLExecute (hstmt);      // insert row

```

The call load has been reduced from 100 to just 2 `SQLExecute` calls; furthermore, network traffic is reduced considerably. Some ODBC drivers do not support `SQLParamOptions`. To achieve high performance, applications should contain algorithms for using `SQLParamOptions` if the ODBC driver supports the function. `SQLParamOptions` is ideal for copying data into new tables or bulk loading tables.

## Using the Cursor Library

Do not automatically use the cursor library if scrollable cursors are provided by the driver. The cursor library creates local temporary log files, which are expensive to generate and provide worse performance than using native scrollable cursors.

The cursor library adds support for static cursors, which simplifies the coding of applications that use scrollable cursors. However, the cursor library creates temporary log files on the user's local disk drive to accomplish the task. Disk I/O is typically one of the slowest operations on personal computers. While the benefits of the cursor library are great, applications should not automatically choose to use the cursor library if an ODBC driver supports scrollable cursors natively.

ODBC drivers that support scrollable cursors typically achieve high performance by requesting that the DBMS server produce a

scrollable result set instead of emulating the capability by creating log files.

Many applications use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_ODBC);
```

but should use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_IF_NEEDED);
```

---

## Design Options

This section provides guidelines for designing ODBC applications.

### Managing Connections

Connection management is important to application performance. Designers should optimize applications by connecting once and using multiple statement handles instead of performing multiple connections. Most ODBC applications contain poorly designed elements for connection management. Connecting to a data source should be avoided after establishing an initial connection.

Some ODBC applications are designed to call informational gathering routines that have no record of already attached connection handles. For example, some applications establish a connection and then call a routine in a separate DLL or shared library that reattaches and gathers information about the driver.

Although gathering driver information at connect time is a good algorithm, it should not be minimized by connecting twice. At least one popular ODBC-enabled application connects a second

time to gather driver information but *never* disconnects the second connection. Applications that are designed as separate entities should pass the already connected HDBC pointer to the data collection routine instead of establishing a second connection.

Another poor practice is to connect and disconnect several times throughout your application to perform SQL statements. Connection handles can have multiple statement handles associated with them. Statement handles are defined to be memory storage for information about SQL statements. Why then do many applications allocate new connection handles to perform SQL statements? Applications should use *statement* handles to manage multiple SQL statements.

Connection and statement handling should not be delayed until implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

## Committing Data

Committing data is extremely disk I/O intensive and slow. Always turn autocommit off, if the driver can support transactions.

What is actually involved in a commit? The database server must flush back to disk every data page that contains updated or new data. Note that this is not a sequential write but a searched write to replace existing data in the table. By default, autocommit is on when connecting to a data source. Autocommit mode is typically detrimental to performance because of the extreme amount of disk I/O needed to commit *every* operation.

To further reduce performance some database servers do not provide an "autocommit mode." For this type of server, the ODBC driver must explicitly issue a COMMIT statement and perhaps a BEGIN TRANSACTION for *every* operation sent to the server. In addition to the large amount of disk I/O required to

support autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

## Asynchronous Execution

Design your application to take advantage of data sources that support asynchronous execution. Asynchronous calls do not perform faster but well-designed applications *appear* to run more efficiently.

By default, an application makes calls to an ODBC driver that then executes statements against the DBMS server in a synchronous manner. In this mode of operation, the driver does not return control to the application until its own request to the server is complete. For statements which take more than a few seconds to complete execution, this can result in the perception of poor performance to the end user.

Some data sources support asynchronous execution. When in asynchronous mode, an application makes calls to an ODBC driver and control is returned almost immediately. In this mode, the driver returns the status `SQL_STILL_EXECUTING` to the application and then sends the appropriate request to the database backend for execution. The application polls the driver at various intervals at which point the driver itself polls the server to see if the query has completed execution. If the query is still executing, the status `SQL_STILL_EXECUTING` is returned to the application. If it has completed, a status such as `SQL_SUCCESS` is returned and the application can then begin to fetch records.

Turning on asynchronous execution does not by itself improve performance. Well-designed applications, however, can take advantage of asynchronous query execution by allowing the end user to work on other things while the query is being evaluated on the server. Perhaps users will start one or more subsequent queries or choose to work in another application, all while the



query is executing on the server. Designing for asynchronous execution makes your application *appear* to run faster by allowing the end user to work concurrently on multiple tasks.

---

## Updating Data

This section provides guidelines for updating data stored in databases with information supplied by the end user.

### Using Positional Updates and Deletes

Use positional updates and deletes or `SQLSetPos` to update data.

Designing an efficient method for updating data is difficult. While positional updates do not apply to all types of applications, developers should attempt to use positional updates and deletes. Positional updates (either through "update where current of cursor" or through `SQLSetPos`) allow the developer to update data simply by positioning the database cursor to the appropriate row to be changed and signal the driver to "change the data here." The designer is not forced to build a complex SQL statement but is simply required to supply the data that is to be changed.

Besides making the application more easily maintainable, positional updates typically result in improved performance. Because the database server is already positioned on the row (for the Select statement in process), performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server typically has an internal pointer to the row available (for example, ROWID).

## Using SQLSpecialColumns

Use `SQLSpecialColumns` to determine the optimal set of columns to use in the `Where` clause for updating data. Many times pseudo-columns provide the fastest access to the data, and these columns can only be determined by using `SQLSpecialColumns`.

Some applications cannot be designed to take advantage of positional updates and deletes. These applications typically update data by forming a `Where` clause consisting of some subset of the column values returned in the result set. Some applications might formulate the `Where` clause by using all searchable result columns or by calling `SQLStatistics` to find columns that might be part of a unique index. These methods typically work, but might result in fairly complex queries.

Consider the following example:

```
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name, ssn, address, city,
    state, zip FROM emp", SQL_NTS);
// fetchdata
...
rc = SQLExecDirect (hstmt, "UPDATE EMP SET ADDRESS = ?
    WHERE first_name = ? and last_name = ? and ssn = ? and address = ? and
    city = ? and state = ? and zip = ?", SQL_NTS);
// fairly complex query
```

Applications should call `SQLSpecialColumns/SQL_BEST_ROWID` to retrieve the most optimal set of columns (possibly a pseudo-column) that identifies any given record. Many databases support special columns that are not explicitly defined by the user in the table definition but are "hidden" columns of every table (for example, `ROWID` and `TID`). These pseudo-columns almost always provide the fastest access to the data because they typically are pointers to the exact location of the record. Because pseudo-columns are not part of the explicit table definition, they are not returned from `SQLColumns`. The only method of determining if pseudo-columns exist is to call `SQLSpecialColumns`.

Consider the previous example again:

```
...
rc = SQLSpecialColumns (hstmt, ..... 'emp', ...);
...
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name, ssn, address, city,
    state, zip, ROWID FROM emp", SQL_NTS);
// fetch data and probably "hide" ROWID from the user
...
rc = SQLExecDirect (hstmt, "UPDATE emp SET address = ? WHERE ROWID = ?",
    SQL_NTS);
// fastest access to the data!
```

If your data source does not contain special pseudo-columns, then the result set of `SQLSpecialColumns` consists of the columns of the most optimal unique index on the specified table (if a unique index exists); therefore, your application need not additionally call `SQLStatistics` to find the smallest unique index.



# G Microsoft Query '97



This Appendix applies only to Windows environments.

---

## Creating a Flat-File Data Source for Use with Microsoft Query '97

To use a flat-file database driver with Microsoft Query '97, you must alter the data source alias outside of Microsoft Query '97. The data source alias created within Microsoft Query sets a default database, which is the Microsoft Query '97 working directory. Most likely, the Microsoft Query '97 working directory does not contain your data files.

The following steps describe how to update a flat-file data source. In this example, a dBASE data file is used:

- 1 Open WordPad or another text editor and edit the data source file that was created in Microsoft Query '97. The file will have a .dsn extension and will be located in the Microsoft Query '97 working directory.

The information will look similar to the section below:

```
[ODBC]
DRIVER={MERANT 3.xx 32-BIT dBASEFile (*.dbf)}
DB=d:\msoffice\query97
```

- 2 Edit the DB entry to specify the directory that contains your data files. For example, the modified file may look similar to:

```
[ODBC]
DRIVER={MERANT 3.xx 32-BIT dBASEFile (*.dbf)}
DB=c:\data\sales
```

- 3 Save the file and exit the editor.

---

## Using Microsoft Query '97 with Single-Connect Data Sources

To use data sources that are limited to a single connection per session (Btrieve and DB2 data sources), you must create the data source as a FileDSN through the ODBC Administrator.

Follow these steps to create a Btrieve DB2 data source. A DB2 data source is used in this example:

- 1 Start the ODBC Administrator.
- 2 Click the **File DSN** tab.
- 3 Click **Add**.
- 4 Select the MERANT 3.xx 32-BIT DB2 data source. Click **Next**.

You are prompted for the path name of the new file data source name.

- 5 Type the Microsoft Query '97 directory followed by your new data source name. For example:

```
d:\msoffice\msquery\DB2Test.dsn
```

Click **Next**.

- 6 Click **Finish** to validate the new File DSN entry.

The driver's logon box is displayed. Specify the logon options as you normally would for the DB2 database. Click **OK**.

- 7 Exit the ODBC Administrator.
- 8 Open WordPad or another text editor and edit the data source file you created in the ODBC Administrator. The file has a .dsn extension and will be located in the Microsoft Query '97 working directory. In our example, the file name is DB2Test.dsn.

The information will look similar to the following example:

```
[ODBC]
DRIVER={MERANT 3.xx 32-BIT DB2}
DB=test
UID=TEST1
```

- 9 Add the following entry to specify the applicable driver option:

```
WA=4
```

The modified file will look similar to the following example:

```
[ODBC]
DRIVER={MERANT 3.xx 32-BIT DB2}
WA=4
DB=test
UID=TEST1
```

- 10 Exit the editor.





# H The UNIX Environments



This appendix contains specific information about using Connect ODBC in the UNIX environments.

---

## The System Information File (.odbc.ini)

In the UNIX environments, there is no ODBC Administrator. To configure a data source, you must edit the system information file, a plain text file that is normally located in the user's \$HOME directory and is usually called *.odbc.ini*. This file is maintained using any text editor to define data source entries as described in the "Connecting to a Data Source Using a Connection String" section of each driver's chapter. You must use the long name of connection string attributes when defining data source entries. A sample file (*odbc.ini*) is located in the driver installation directory.

UNIX support of the database drivers also permits the use of a centralized system information file that a system administrator can control. This is accomplished by setting the environment variable `ODBCINI` to point to the fully qualified pathname of the centralized file. For example, in the C shell you could set this variable as follows:

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

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

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

The search order for the location of the system information file is as follows:

- 1 Check ODBCINI
- 2 Check \$HOME for .odbc.ini

There must be an [ODBC] section in the system information file that includes the InstallDir keyword. The value of this keyword must be the path to the directory under which the /lib and /messages directories are contained. For example, if you choose the default install directory, then the following line must be in the [ODBC] section:

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

## Sample Solaris System Information File

In the following sample, xx represents the driver number:

```
[ODBC Data Sources]
DB2_MVS=Sample DB2 DRDA dsn
DB2_AIX=Sample DB2 DRDA dsn
dBase=Sample dBASE dsn
Informix=Sample Informix dsn
Informix_CL=Sample Informix Dynamic Server dsn
MS_SQLServer7=Sample MS SQLServer dsn
Oracle7=Sample Oracle7 dsn
Oracle8=Sample Oracle8 dsn
Sybase=Sample Sybase dsn
SybaseASE=Sample Sybase ASE dsn
Text=Sample Text file dsn

[DB2_MVS]
Driver=/opt/odbc/lib/ivdb2xx.so
Description=DRDA driver
LogonID=TEST
Location=MVSDDB2
Collection=DATADIRECT00
Package=DEF00
```

```
IPAddress=mvsmachine  
TcpPort=5179  
AddStringToCreateTable=in database ovsusa
```

```
[DB2_AIX]  
Driver=/opt/odbc/lib/ivdb2xx.so  
Description=DRDA driver to DB2 v6.1 AIX  
LogonID=TEST  
Database=TEST  
Package=DEF00  
IPAddress=10.10.10.10  
TcpPort=50002
```

```
[dBase]  
Driver=/opt/odbc/lib/ivdbfxx.so  
Description=dBase  
Database=/opt/odbc/demo
```

```
[Informix]  
Driver=/opt/odbc/lib/ivinfxx.so  
Description=Informix  
Database=odbc  
HostName=informixhost  
LogonID=odbc01  
Password=odbc01
```

```
[Informix_CL]  
Driver=/opt/odbc/lib/ivifclxx.so  
Description=Informix_CL  
Database=test  
LogonID=test  
ServerName=pilot920  
HostName=pilot  
PortNumber=2003
```

```
[MS_SQLServer7]  
Driver=/opt/odbc/lib/ivmsssxx.so  
Description=SQL Server 7  
Database=test  
LogonID=test  
Address=orca,1433
```

```
[Oracle7]
Driver=/opt/odbc/lib/ivor7xx.so
Description=Oracle7
LogonID=test
ServerName=orcaTEST
```

```
[Oracle8]
Driver=/opt/odbc/lib/ivor8xx.so
Description=Oracle8
ServerName=oraclehost
LogonID=odbc01
Password=odbc01
```

```
[Sybase]
Driver=/opt/odbc/lib/ivsybxx.so
Description=Sybase
Database=odbc
ServerName=SYBASE
LogonID=odbc01
Password=odbc01
OptimizePrepare=2
SelectMethod=1
```

```
[SybaseASE]
Driver=/opt/odbc/lib/ivasexx.so
Description=Sybase
Database=test
LogonID=test
NetworkAddress=10.30.14.72,5000
```

```
[Text]
Driver=/opt/odbc/lib/ivtxtxx.so
Description=Text driver
Database=/opt/odbc/demo
```

```
[ODBC]
Trace=0
TraceFile=odbctrace.out
TraceDll=/opt/odbc/lib/odbctrac.so
InstallDir=/opt/odbc
```

---

# Environment Variables

Connect ODBC drivers require several environment variables to be set.

## Required Environment Variables

Most of the variables can be set by executing the appropriate shell script located in the ODBC home directory.

For example, C shell (and related shell) users should execute the following command before attempting to use ODBC-enabled applications:

```
% source odbc.csh
```

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

```
$ . odbc.sh
```

Executing these scripts will set the appropriate library search path environment variable (LD\_LIBRARY\_PATH on Solaris and Linux, SHLIB\_PATH on HP/UX, or LIBPATH on AIX).

The library search path environment variables are required to be set so that the ODBC core components and drivers can be located at the time of execution.

## Optional Environment Variables

Many of the Connect ODBC drivers must have environment variables set as required by the database client components used by the drivers. Consult the driver requirements in each of the individual driver sections for additional information pertaining to individual driver requirements.

ODBCINI is an optional environment variable that all Connect ODBC drivers will recognize. ODBCINI is used to locate an ODBC information file other than the default file and is described in detail under [“The System Information File \(.odbc.ini\)”](#) on [page 473](#).

---

## Using Double-Byte Character Sets

Connect ODBC drivers are capable of using double-byte character sets. The drivers normally use the character set defined by the default locale "C" unless explicitly pointed to another character set. The default locale "C" corresponds to the 7-bit ASCII character set in which only characters from ISO 8859-1 are valid. Use the following procedure to set the locale to a different character set.

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

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

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

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

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

## The ivtestlib Tool

The ivtestlib tool is provided to help diagnose configuration problems (such as environment variables not correctly set or missing DBMS client components) in the UNIX environment. This command will attempt to load a specified ODBC driver and will print out all available error information if the load fails.

On HP-UX, for example, if a driver is installed in /opt/odbc, the command:

```
ivtestlib /opt/odbc/lib/ivinfxx.sl
```

(where *xx* represents the driver number) will attempt to load the Informix driver. If the driver cannot be loaded, ivtestlib will return an error message explaining why.

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

---

## Translators

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

You must add the TranslationSharedLibrary keyword to the data source section of the system information file to perform a translation. Adding the TranslationOption keyword is optional.

<b>Keyword</b>	<b>Definition</b>
TranslationSharedLibrary	Full path of translation library
TranslationOption	ASCII representation of the 32-bit integer translation option



# I Values for AppCodePage Connection String Attribute



[Table I-1](#) lists valid values for the AppCodePage connection string attribute. This attribute is valid only for Connect ODBC drivers that run on UNIX. See the appropriate individual driver chapter for information about this attribute.

---

**Table I-1. AppCodePage Values**

---

Value	Description
0	ISO 646 7-bit ASCII
1	ISO 8859-1 Latin-1
2	CP 850 - European code page
3	CP 437 - US code page
4	HP ROMAN8
5	Standard Macintosh Roman
6	Shift-JIS proper
7	EUC-JIS encoding
8	Digital UNIX JIS encoding
9	EUC-CNS encoding
10	EUC-GB encoding
11	Microsoft CP 932 = Win32J-DBCS
12	ISO 8859-2 Latin-2 Eastern Europe
13	ISO 8859-5 Latin/Cyrillic
14	ISO 8859-6 Latin/Arabic
15	ISO 8859-7 Latin/Greek
16	ISO 8859-8 Latin/Hebrew
17	ISO 8859-9 Latin-5 Turkish

**Table I-1. AppCodePage Values** (cont.)

<b>Value</b>	<b>Description</b>
18	ISO 8859-10 Latin-6 Nordic
19	ISO 8859-3 Latin/Esperanto/Galician
20	ISO 8859-4 Latin/Estonian/Latvian
21	ISO 8859-15 Latin-9 Western Europe with Euro sign
26	Macintosh Cyrillic
27	Macintosh Eastern European
28	Macintosh Greek
29	Macintosh Turkish
30	HP Greek
31	HP Turkish
32	KOI8 - Cyrillic
33	TIS 620 - Thai standard
34	Big5 Traditional Chinese
36	EUC-KSC Korean encoding, similar to CP 949
37	IBM EBCDIC (8859-1 convertible)
273	IBM EBCDIC Germany/Austria
277	IBM EBCDIC Denmark/Norway
278	IBM EBCDIC Finland/Sweden
280	IBM EBCDIC Italian
284	IBM EBCDIC Spain/Latin America
285	IBM EBCDIC U.K.
290	IBM EBCDIC Katakana for DB2
297	IBM EBCDIC France
420	IBM EBCDIC Arabic bilingual
500	IBM EBCDIC Western Europe
737	PC Greek

**Table I-1. AppCodePage Values** (cont.)

<b>Value</b>	<b>Description</b>
775	PC Baltic
852	PC Eastern Europe
855	PC Cyrillic
857	PC Turkish
860	PC Portuguese
861	PC Icelandic
862	PC Hebrew
863	PC Canadian French
864	PC Arabic
865	PC Nordic
866	PC Russian
869	PC Greek
870	IBM EBCDIC Eastern Europe
874	Microsoft Thai SB code page
875	IBM EBCDIC Greek
930	Japanese Host merged: CP 290 + CP 300
932	Japanese IBM J-DBCS: CP 897 + CP 301
933	Korean Host merged: CP 833 + CP 834
935	SimpChinese Host merged: CP 836+ CP 837
936	PC Simplified Chinese
937	TradChinese Host merged: CP 037 + CP 835
939	Japanese Host merged: CP 1027 + CP 4396
949	PC (MS) Korean, similar to EUC-KSC
950	PC (MS) Traditional Chinese (~Big5)
954	EUC-JIS
1026	IBM EBCDIC Turkish
1047	MVS Open Edition
1250	MS Windows 3.1 Eastern European

---

**Table I-1. AppCodePage Values** (cont.)

---

<b>Value</b>	<b>Description</b>
1251	MS Windows 3.1 Cyrillic
1252	MS Windows 3.1 US (ANSI)
1253	MS Windows 3.1 Greek
1254	MS Windows 3.1 Turkish
1255	MS Windows Hebrew
1256	MS Windows Arabic
1257	MS Windows Baltic
1258	MS Windows Vietnamese
5026	CCSID for CP 930 with only 1880 UDC
5035	CCSID for CP 939 with only 1880 UDC

---

# Index

## Symbols

# symbol in regular identifiers 383

## A

ADO 2.5 persisted file format for XML driver 361

aggregate functions, flat-file drivers 394

AIX 34

aliasing table references 382

Alter Table statement

    Btrieve 61

    dBASE 118

    Paradox 216

    Text 357

application code page. *See* code pages

## B

Btrieve driver

*See also* flat-file drivers

    Alter Table statement 61

    column names 60

    connection string attributes 55

    connections supported 64

    Create Index statement 62

    data dictionary file 45

    data source

        configuring 46

        connecting via connection string 54

    data types 58

    driver requirements 44

Drop Index statement 62

indexes 59

isolation levels 63

locking levels 63

managing databases 45

ODBC conformance 63

Rowid pseudo-column 60

Scalable SQL 45

Select statement 60

statements supported 64

table structure 52

transactions 45

## C

character string literals 384

client code page. *See* code pages

code pages

    connection string attribute 481

    DB2 driver 80

    dBASE driver 110

    Informix driver 149

    Informix Dynamic Server driver 166

    MySQL driver 174

    Oracle driver 189

    SQL Server driver 270

    Sybase ASE driver 320

    Text driver 352

comments, SQL 391

configuring data source. *See* data source, configuring

Connect ODBC

    Btrieve 43

    DB2 65

    dBASE 93

    Excel Workbook 127

- Informix 139
- Informix Dynamic Server 159
- MySQL 173
- Oracle 177
- Paradox 203
- PROGRESS 229
- SQL Server 255
- SQL Server 6 277
- SQLBase 295
- Sybase ASE 307
- Text 333
- XML 359
- connecting to data source. *See* data source, connecting
- connection string attributes
  - Btrieve 55
  - DB2 80
  - dBASE 110
  - Excel Workbook 134
  - Informix 149
  - Informix Dynamic Server 166
  - MySQL 174
  - Oracle 189
  - Paradox driver 211
  - PROGRESS 251
  - SQL Server (UNIX) 269
  - SQL Server (Windows) 263
  - SQL Server 6 287
  - SQLBase 302
  - Sybase ASE 320
  - Text 352
  - XML 369
- connections supported
  - Btrieve 64
  - DB2 92
  - dBASE 126
  - Excel Workbook 137
  - Informix 158, 172
  - MySQL 176
  - Oracle 201
  - Paradox 227
  - PROGRESS 254
  - SQL Server 275
  - SQL Server 6 293

- SQLBase 305
- Sybase ASE 332
- Text 358
- XML 381
- contacting Technical Support 29
- conventions, typographical 24
- Create Index statement
  - Btrieve driver 62
  - dBASE 119
  - Paradox 224
- Create Table statement
  - flat-file drivers 411
  - Paradox 217

## D

- Data Island format for XML driver 361
- data source
  - configuring
    - Btrieve 46
    - DB2 66
    - dBASE 94
    - Excel Workbook 128
    - FoxPro DBC 100
    - Informix 141
    - Informix Dynamic Server 159
    - MySQL 173
    - Oracle 181
    - Paradox 205
    - PROGRESS 230
    - SQL Server 256
    - SQL Server 6 278
    - SQLBase 296
    - Sybase ASE 307
    - Text 335
    - XML 362
  - connecting via connection string
    - Btrieve 54
    - DB2 79
    - dBASE driver 109
    - Excel Workbook 133
    - Informix 148

- Informix Dynamic Server 165
- MySQL 173
- Oracle 188
- Paradox 210
- PROGRESS 250
- SQL Server 262
- SQL Server 6 286
- SQLBase 301
- Sybase ASE 319
- Text 351
- XML 368
- connecting via logon dialog box
  - DB2 77
  - Informix 147
  - Informix Dynamic Server 164
  - Oracle 187
  - PROGRESS 247
  - SQL Server 260
  - SQL Server 6 284
  - SQLBase 300
  - Sybase ASE 318
  - XML 367
- data types
  - Btrieve 58
  - DB2 86
  - dBASE 114
  - Excel Workbook 136
  - Informix 152
  - Informix Dynamic Server 168
  - MySQL 175
  - Oracle 193
  - Oracle8 194
  - Paradox 214
  - PROGRESS 253
  - SQL Server 273
  - SQL Server 6 290
  - SQLBase 303
  - Sybase ASE 326
  - Text 356
  - XML 375
- date and time functions 435
- date and time literals 385
- date masks, defining for the Text driver 349
- DB2 driver
  - code pages 80
  - connection string attributes 80
  - connections supported 92
  - data source
    - configuring 66
    - connecting via connection string 79
    - connecting via logon dialog box 77
  - data types 86
  - driver requirements 65
  - isolation levels 91
  - locking levels 91
  - ODBC conformance 91
  - persisting result set as XML 89
  - statements supported 92
  - stored procedure support 91
  - Unicode support 87
- dBASE driver
  - See *also* flat-file drivers
  - Alter Table statement 118
  - code pages 110
  - column names 116
  - connection string attributes 110
  - connections supported 126
  - Create Index statement 119
  - data source
    - configuring 94
    - connecting via connection string 109
  - data types 114
  - defining index attributes 106
  - defining index attributes under UNIX 108
  - driver requirements 93
  - Drop Index statement 121
  - isolation levels 126
  - locking 124
  - locking levels 126
  - ODBC conformance 126
  - Pack statement 122
  - Rowid pseudo-column 117
  - Select statement 116
  - statements supported 126
- Delete statement, flat-file drivers 416
- delimited identifiers 383
- distributed transactions. See MTS support

- documentation
  - about 26
  - order form 28
  - ordering printed books 27
- double-byte character sets in UNIX 478
- driver requirements
  - Btrieve driver 44
  - DB2 driver 65
  - dBASE driver 93
  - Excel Workbook driver 127
  - Informix driver 139
  - Informix Dynamic Server driver 159
  - MySQL 173
  - MySQL driver 173
  - Oracle driver 177
  - Paradox driver 203
  - PROGRESS driver 229
  - SQL Server 6 driver 277
  - SQL Server driver 255
  - SQLBase driver 295
  - Sybase ASE driver 307
  - Text driver 333
  - XML driver 360
- drivers, Connect ODBC
  - about 31
  - See *also* Connect ODBC
- Drop Index statement
  - Btrieve driver 62
  - dBASE driver 121
  - Paradox 225
- Drop Table statement, flat-file drivers 412

## E

- environment-specific information 25, 33
- error messages
  - general 40
  - UNIX 41
- escape sequences 386

- Excel Workbook driver
  - See *also* flat-file drivers
  - column names 136
  - connection string attributes 134
  - connections supported 137
  - data source
    - configuring 128
    - connecting via connection string 133
  - data types 136
  - driver requirements 127
  - Excel Workbook database 127
  - ODBC conformance 137
  - SQL supported 136
  - statements supported 137
  - table names 136
- extensions to SQL standards 382

## F

- flat-file drivers
  - aggregate functions 394
  - Create Table statement 411
  - Delete statement 416
  - Drop Table statement 412
  - For Update clause 399
  - From clause 395
  - Group By 397
  - Having clause 397
  - Insert statement 413
  - operator precedence 405
  - Order By 399
  - Select clause 394
  - Select statement 393
  - SQL 393
  - SQL expressions 400
  - Union operator 398
  - Update statement 415
  - Where clause 396
- For Update clause, flat-file drivers 399
- formats, for text files 334



FoxPro  
   configuring DBC data source 100  
   SQL statements 123  
 From clause, flat-file drivers 395

## G

getting started 31  
 grammar, tokens used in SQL 382  
 Group By clause, flat-file drivers 397  
 GUID literal 385

## H

HA Failover server, defining 309  
 Having clause, flat-file drivers 397  
 help, online 26  
 hex literals 385  
 hints, using with XML driver 372  
 HP-UX aCC 35

## I

identifiers  
   # symbol in regular 383  
   delimited 383  
   regular 383  
 improving  
   database performance 419  
   index performance 419  
   join performance 425  
   ODBC applications performance 449  
   record selection performance 421  
 indexes  
   deciding which to create 423  
   improving performance 419  
 indexing multiple fields 421

Informix driver  
   code pages 149  
   connection string attributes 149  
   connections supported 158  
   data source  
     configuring 141  
     connecting via connection string 148  
     connecting via logon dialog box 147  
   data types 152  
   driver requirements 139  
   enabling MTS 155  
   isolation levels 157  
   locking levels 157  
   MTS support 155  
   ODBC conformance 158  
   persisting result set as XML 156  
   statements supported 158  
 Informix Dynamic Server driver  
   code pages 166  
   connection string attributes 166  
   connections supported 172  
   data source  
     configuring 159  
     connecting via connection string 165  
     connecting via logon dialog box 164  
   data types 168  
   driver requirements 159  
   enabling MTS 167  
   isolation levels 171  
   locking levels 171  
   ODBC conformance 172  
   persisting result set as XML 170  
   SQL grammar supported 172  
   statements supported 172  
 Insert statement  
   flat-file drivers 413  
   XML driver 381  
 integer numbers 384  
 interoperability 31  
 isolation levels  
   Btrieve 63  
   DB2 91  
   dBASE 126  
   Informix 157

- Informix Dynamic Server 171
- Oracle 200
- Paradox 226
- PROGRESS 235, 253
- SQL Server 274
- SQL Server 6 292
- SQLBase 304
- Sybase ASE 332
- isolation levels and data consistency
  - compared 442
  - dirty reads 440
  - non-repeatable reads 440
  - phantom reads 440
- isolation levels, general 440
- isolation levels, specific
  - read committed 441
  - read uncommitted 441
  - repeatable read 441
  - serializable 441
- ivtestlib tool 479

## J

- joins 382

## K

- keywords, reserved 386

## L

- Left Outer Joins 382
- library path environment variable 38
- Linux 36
- literal tokens 385

- literals
  - character string 384
  - date and time 385
  - GUID 385
  - hex 385
  - ODBC time and date 385
  - time 385
  - timestamp 385
- locking levels
  - Btrieve 63
  - DB2 91
  - dBASE 126
  - Informix 157
  - Informix Dynamic Server 171
  - Oracle 200
  - Paradox 204, 226
  - PROGRESS 253
  - SQL Server 274
  - SQL Server 6 292
  - SQLBase 304
  - Sybase ASE 332
- locking modes and levels 443

## M

- Microsoft Data Island format for XML driver 361
- MTS support
  - Informix driver 155
  - Informix Dynamic Server driver 167
  - Oracle8 driver 199
  - Sybase ASE driver 311, 327
- MySQL driver
  - code pages 174
  - connection string attributes 174
  - connections supported 176
  - data source
    - configuring 173
    - connecting via connection string 173
  - data types 175
  - driver requirements 173

ODBC conformance 176  
statements supported 176

## N

numbers, integer 384  
numbers, real 384  
numeric functions 433

## O

ODBC  
API functions 427  
designing for performance 449  
scalar functions 430  
specification 31  
time and date literals 385

ODBC conformance  
Btrieve 63  
DB2 91  
dBASE 126  
Excel Workbook 137  
Informix 158  
Informix Dynamic Server 172  
MySQL 176  
Oracle 200  
Paradox 227  
PROGRESS driver 254  
SQL Server 274  
SQL Server 6 293  
SQLBase 305  
Sybase ASE 332  
Text 358  
XML 380

ODBCINI 478  
online books, order form 28  
Open Database Connectivity. *See* ODBC  
Oracle driver  
code pages 189  
connection string attributes 189

connections supported 201  
data source  
configuring 181  
connecting via connection string 188  
connecting via logon dialog box 187  
data types 193  
driver requirements 177  
isolation levels 200  
locking levels 200  
ODBC conformance 200  
persisting result set as XML 197  
SQL grammar supported 176  
statements supported 201  
threading 185  
Unicode support 195

Oracle8 driver  
data types 194  
MTS support 199  
*See also* Oracle driver

Order By clause, flat-file drivers 399  
ordering printed books 27  
Outer Joins 382

## P

Pack statement, using with dBASE driver 122  
packet size  
setting for Oracle7 driver 184  
setting for Sybase ASE driver 315, 323  
SQL Server 6 282

Paradox driver  
*See also* flat-file drivers  
Alter Table statement 216  
connection string attributes 211  
connections supported 227  
Create Index Primary statement 223  
Create Index statement 224  
Create Table statement 217  
data source  
configuring 205  
connecting via connection string 210  
data types 214

- decrypting table 220
- driver requirements 203
- Drop Index statement 225
- dropping columns 217
- encrypting table 219
- encryption 218
- index files 221
- isolation levels 226
- locking levels 204, 226
- ODBC conformance 227
- passwords 218
  - removing 220
- Select statement 216
- statements supported 227
- table access, multiuser 204
- transactions 226
- passwords
  - Paradox driver 218
  - removing 220
- performance, improving
  - database 419
  - index 419
  - join 425
  - ODBC applications 449
  - record selection 421
- persisting result set as XML
  - DB2 driver 89
  - Informix driver 156
  - Informix Dynamic Server 170
  - Oracle driver 197
  - SQL Server 6 driver 291
  - Sybase ASE driver 330
  - XML driver 379
- Pervasive.SQL. See Btrieve driver
- printed books, order form 28
- PROGRESS driver
  - configuring the environment 243
  - connection string attributes 251
  - connections supported 254
  - data source
    - configuring 230
    - connecting via connection string 250
    - connecting via logon dialog box 247
  - data types 253

- driver requirements 229
- isolation levels 253
- locking levels 253
- ODBC conformance 254
- OID with access via server 237
- OID with direct access 230
- setting system variables 244
- statements supported 254

## Q

- query timeout, setting for the Sybase ASE driver 331

## R

- real numbers 384
- references, aliasing table 382
- regular identifiers 383
- reserved keywords for XML driver 386
- reserved words 417
- row hints for XML documents 372
- Rowid pseudo-column
  - Btrieve driver 60
  - dBASE 117

## S

- scalar functions, ODBC 430
- schema
  - deleting externally linked files 365
  - format supported by XML driver 361
  - validating an XML document 364
- Select clause, flat-file drivers 394
- Select statement
  - dBASE driver 116
  - Excel Workbook driver 136

- flat-file drivers 393
    - Paradox 216
    - Text 357
    - XML driver 381
  - Solaris 36
  - sorting
    - dBASE driver 98, 104
    - Excel Workbook driver 135
    - Text driver 340
    - XML driver 365, 369
  - SQL
    - comments 391
    - expressions, flat-file drivers 400
    - extensions to ANSI SQL 92 382
    - flat-file drivers 393
    - grammar token definitions 382
    - grammar, tokens used in 382
    - operators and symbols 386
    - reserved words 417
    - standards, extensions to 382
    - syntax for date and time literals 385
  - SQL Server 6 driver
    - connection string attributes 287
    - connections supported 293
    - data source
      - configuring 278
      - connecting via connection string 286
      - connecting via logon dialog box 284
    - data types 290
    - driver requirements 277
    - isolation levels 292
    - locking levels 292
    - ODBC conformance 293
    - persisting result set as XML 291
    - SQL grammar support 293
    - statements supported 293
  - SQL Server driver
    - code pages 270
    - connection string attributes (UNIX) 269
    - connection string attributes (Windows) 263
    - connections supported 275
  - data source
    - configuring 256
    - connecting via connection string 262
    - connecting via logon dialog box 260
  - data types 273
  - driver requirements 255
  - isolation levels 274
  - locking levels 274
  - ODBC conformance 274
  - statements supported 275
  - unicode support 272
- SQLBase driver
    - connection string attributes 302
    - connections supported 305
    - data source
      - configuring 296
      - connecting via connection string 301
      - connecting via logon dialog box 300
    - data types 303
    - driver requirements 295
    - isolation levels 304
    - locking levels 304
    - ODBC conformance 305
    - statements supported 305
  - standards
    - compliance to ODBC specification 31
    - extensions to SQL 382
  - statements supported
    - Btrieve 64
    - DB2 92
    - dBASE 126
    - Excel Workbook 137
    - Informix 158
    - Informix Dynamic Server 172
    - MySQL 176
    - Oracle 201
    - Paradox 227
    - PROGRESS 254
    - SQL Server 275
    - SQL Server 6 293
    - SQLBase 305
    - Sybase ASE 332
    - Text 358
    - XML 381

- stored procedures
  - creating for Sybase ASE driver 323
  - DB2 driver 91
  - Oracle 185
- stored results, Oracle driver 199
- string functions 430
- string literals, character 384
- Structured Query Language. *See* SQL
- SupportNet 29
- Sybase ASE driver
  - code pages 320
  - connection string attributes 320
  - connections supported 332
  - data source
    - configuring 307
    - connecting via connection string 319
    - connecting via logon dialog box 318
  - data types 326
  - driver requirements 307
  - isolation levels 332
  - locking levels 332
  - model for distributed transaction support 311
  - MTS support 327
  - ODBC conformance 332
  - persisting result set as XML 330
  - query timeout 331
  - statements supported 332
  - Unicode support 327
- symbol in regular identifiers, # 383
- symbols, SQL 386
- system functions 437
- system information file (.odbc.ini) 473
- system requirements. *See* driver requirements

## T

- table hints for XML documents 372
- table references, aliasing 382
- table structure, defining
  - Btrieve 52
  - Text 342
  - Text under UNIX 346
- Technical Support, contacting 29
- Text driver
  - See also* flat-file drivers
  - Alter Table statement 357
  - code pages 352
  - connection string attributes 352
  - connections supported 358
  - data source
    - configuring 335
    - connecting via connection string 351
  - data types 356
  - date masks 349
  - defining table structure 342
  - defining table structure under UNIX 346
  - driver requirements 333
  - formats 334
  - ODBC conformance 358
  - Select statement 357
  - sort order 340
  - statements supported 358
- threading
  - dBASE driver 99
  - Excel Workbook driver 131
  - Informix driver 145
  - Informix Dynamic Server driver 162, 166
  - Oracle driver 185
  - overview 445
  - Paradox driver 208, 211
  - SQL Server 6 283
  - SQLBase driver 299, 302
  - Sybase driver 311
  - Text driver 341, 352
- time functions 435
- time literals 385
- timestamp literals 385
- tokens
  - literal 385
  - used in SQL grammar 382

transactions  
 Btrieve driver 45  
 effect of record locks with dBASE driver 125  
 Paradox driver 226  
 Sybase ASE driver 312

translator  
 in the UNIX environment 480  
 See configuration procedure in Windows driver chapters

typographical conventions 24

## U

Unicode support  
 DB2 driver 87  
 Oracle driver 195  
 SQL Server driver 272  
 Sybase ASE driver 314, 327

Union operator, flat-file drivers 398

UNIX  
 code page connection string attributes 481  
 driver names 37  
 drivers  
 dBASE 93  
 Informix 139  
 Informix Dynamic Server 159  
 MySQL 173  
 Oracle 177  
 SQL Server 255  
 Sybase ASE 307  
 Text 333

environment  
 double-byte character sets 478  
 introduction 473  
 ivtestlib tool 479  
 library path 38  
 system information file (.odbc.ini) 37, 473  
 translators 480  
 variables 477

error messages 41  
 system requirements 34

Update statement  
 flat-file drivers 415  
 XML driver 381

used in SQL grammar, tokens 382

## V

validating XML document against its schema 364

## W

Where clause, flat-file drivers 396

Windows  
 driver names 33  
 drivers  
 Btrieve 43  
 dBASE 93  
 Excel Workbook 127  
 Informix 139  
 Informix Dynamic Server 159  
 Oracle 177  
 Paradox 203  
 PROGRESS 229  
 SQL Server 255  
 SQL Server 6 277  
 SQLBase 295  
 Sybase ASE 307  
 Text 333  
 XML 359

starting the ODBC Administrator 33  
 system requirements 34

## X

### XML driver

- connection string attributes 369
- connections supported 381
- data source
  - configuring 362
  - connecting via connection string 368
  - connecting via logon dialog box 367
- data types 375
- defining style for new tables 369
- driver requirements 360
- formats 361
- ODBC conformance 380
- persisting result set as XML 379
- reserved keywords 386
- sorting order 365
- SQL statements supported 359
- statements supported 381
- terminology differences 360
- using hints 372
- validating an XML document against its schema 364

XML persistence. See persisting result set as XML