



SequeLink[®]

Developer's Reference

Release 6.0
April 2008

© 2008 Progress Software Corporation. All rights reserved. Printed in the U.S.A.

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

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

DataDirect products for the Microsoft SQL Server database:

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

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

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

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

DataDirect SequeLink includes:

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

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

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

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

Table of Contents

List of Tables	15
Preface	19
What Is DataDirect SequeLink®?	19
Using This Book	19
Other SequeLink® Documentation	22
HTML Version	22
PDF Version	24
Typographical Conventions	25
Environment-Specific Information	26
Contacting Technical Support	27
 Part 1: Developing ODBC Applications	
1 Using the ODBC Client	31
About the ODBC Client	31
Using the ODBC Administrator	32
Configuring ODBC Client Data Sources on Windows	33
Configuring ODBC User and System Client Data Sources	33
Configuring ODBC File Client Data Sources	42
ODBC Connection Dialogs	46
Testing ODBC Connections on Windows	52
Configuring ODBC Client Data Sources on Linux and UNIX	53
Configuring the System Information File	53
Example: odbc.ini for Solaris	54

Example: odbc64.ini for Solaris	54
Setting Environment Variables	55
Using a Centralized System Information File	56
Connecting Using a Connection String	57
DSN-less Connections in Linux and UNIX	58
ODBC Connection Attributes.	60
Configuring Connection Failover	76
Connection Failover Properties	79
Using Client Load Balancing	81
Using Connection Retry	82
2 Developing ODBC Applications	83
Required ODBC Libraries and Header Files.	84
Compiler Requirements	84
ODBC API Functions	85
Binding SQL Statements.	88
Support for Unicode ODBC W (Wide) Function Calls. . . .	89
SQL Escape Sequences	90
Data Types and Isolation Levels.	91
Threading	91
Threading Architecture	91
Cancelling Functions in Multithreaded Applications. . . .	92
Using Scrollable Cursors.	93
Static and Keyset-Driven Cursors.	93
Using Static Scrollable Cursors.	94
Using Keyset-Driven Scrollable Cursors.	95
Using Stored Procedures with Oracle	95
Specifying Application IDs	99
Specifying Application IDs Explicitly	99
Generating Application IDs Automatically.	100

Sending Arrays of Parameters	101
Persisting a Result Set as an XML Data File	101
Error Handling	103
SequeLink® for ODBC Driver Errors	103
SequeLink® Client Errors	103
SequeLink® Server Errors	104
Database Errors	104
Developing Performance-Optimized ODBC Applications . . .	105
Catalog Functions	105
Retrieving Data	110
Selecting ODBC Function	115
Managing Connections and Updates	120

Part 2: Developing ADO Applications

3 Using the ADO Client	127
About the ADO Client	127
Using the DataDirect Configuration Manager	128
Working with the DataDirect Configuration Manager . .	130
Displaying Data Source Properties	131
Configuring ADO Client Data Sources	133
Creating an ADO Client Data Source	133
Modifying an ADO Client Data Source	140
Renaming an ADO Client Data Source	141
Deleting an ADO Client Data Source	141
Copying an ADO Client Data Source	142
Changing Data Source Directories	143
Defining Default Setup Options	143
Connecting to an ADO Client	147
Testing ADO Connections	147
ADO Connection Dialogs	148
Connecting with a Provider String	155
ADO Connection Attributes	156

Configuring Connection Failover	164
Connection Failover Properties	167
Using Client Load Balancing	168
Using Connection Retry	169
4 Developing ADO Applications	171
OLE DB Objects and Interfaces	172
Supported Schema Rowsets	174
Supported OLE DB Property Groups	175
Data Source Property Group	176
Data Source Information Property Group	176
Initialization Property Group	182
Rowset Property Group	183
Session Property Group	188
OLE DB Interfaces Supported in ADO	188
Mapping ADO Methods and Properties	190
ADO Command Object	190
Connection Object	193
Recordset Object	201
Data Shaping	206
Persisting Information	207
Using Rowsets	207
Mapping Data Types	208
Specifying Application IDs	208
Specifying Application IDs Explicitly	208
Generating Application IDs Automatically	209
Error Handling	209
SequeLink® for ADO Provider Errors	210
SequeLink® Client Errors	210
SequeLink® Server Errors	211
Database Errors	211

Part 3: Developing JDBC Applications

5	Using the JDBC Client	215
	About the JDBC Client.	215
	JDBC Driver.	216
	SequeLink® Proxy Server	216
	DataDirect Spy™	217
	DataDirect Test™	218
	DataDirect Connection Pool Manager	218
	J2EE Connector Architecture (JCA) Resource Adapter	219
	JDBC Client Directory Structure	220
	Registering the JDBC Driver	223
	Specifying JDBC Driver Connection URLs.	224
	Configuring JDBC Data Sources	225
	Creating and Managing JDBC Data Sources	226
	Calling a Data Source in an Application.	227
	Using JNDI for Naming Databases	228
	Using Connection Pooling.	229
	Using the Java Transaction API.	231
	J2EE Connector Architecture Resource Adapter	232
	Using the Resource Adapter with an Application Server	233
	Using the Resource Adapter from an Application.	233
	Specifying Connection Properties.	235
	Using Connection URLs or the JDBC Driver Manager	236
	Using JDBC Data Sources.	236
	JDBC Connection Properties	237
	Configuring Connection Failover	254
	Using Client Load Balancing	257
	Using Connection Retry.	258

Testing JDBC Connections	258
Using the JDBC Client on a Java 2 Platform	259
6 Using DataDirect Test™	263
DataDirect Test™ Tutorial	263
Configuring DataDirect Test™	264
Starting DataDirect Test™	265
Connecting Using DataDirect Test™	267
Executing a Simple Select Statement	272
Executing a Prepared Statement	274
Retrieving Database Metadata	278
Scrolling Through a Result Set	281
Batch Execution on a Prepared Statement	284
Returning ParameterMetaData	288
Establishing Savepoints	290
Updatable Result Sets	297
LOB Support	309
7 Tracking JDBC Calls	315
About DataDirect Spy™	315
Enabling DataDirect Spy™	316
Using the JDBC Driver Manager	317
Using JDBC Data Sources	318
Using the DataDirect Spy™ URL	320
Registering the DataDirect Spy™ JDBC Driver	322
DataDirect Spy™ Attributes	323
Using DataDirect Spy™ with JDBC Data Sources	324
Checking the DataDirect Spy™ Version	325

8	Developing JDBC Applications	327
	JDBC 3.0 Support	328
	JCA Resource Adapter Class	329
	SQL Support	329
	Binding SQL Statements	329
	Data Types and Isolation Levels	331
	Threading	331
	Threading Architecture	331
	Cancelling Functions in Multithreaded Applications	332
	Using Scrollable Cursors	333
	Result Set Types	333
	Concurrency Types	334
	Using Scrollable Cursors	335
	Specifying Application IDs	336
	Parameter Metadata Support	337
	INSERT and UPDATE Statements	337
	Select Statements	338
	ResultSet Metadata Support	339
	Unicode Support	341
	Rowset Support	341
	Error Handling	342
	Driver Errors	342
	SequeLink® Server Errors	343
	Database Errors	343
	Fine-Tuning JDBC Application Performance	344
	Reducing Download Time	344
	Fetching BigDecimal Objects	346
	Using Database Metadata Methods	346
	Retrieving Data	349
	Selecting JDBC Objects and Methods	351
	Using get Methods Effectively	354

Designing JDBC Applications. 355

Updating Data 357

Part 4: Developing .NET Applications

9 Using the .NET Client 363

About the .NET Client 363

Using Connection Pooling 364

 Creating a Connection Pool. 364

 Adding Connections to a Pool. 365

 Removing Connections from a Pool 366

 Handling Dead Connection in a Pool 367

 Handling Distributed Transactions in a Pool. 368

 Tracking Connection Pool Performance 369

Specifying Connection Options 369

Using Connection Failover. 375

Client Load Balancing 377

Using .NET Objects 378

Assemblies. 378

Parameter Markers. 379

Parameter Arrays 379

Stored Procedures 380

Transaction Support 381

 Using Local Transactions 381

 Using Distributed Transactions 381

Connecting to a Database 383

10 Developing .NET Applications	389
Namespace	389
Data Types	390
Mapping Parameter Data Types	392
Isolation Levels	395
Threading	395
Event Handling	396
.NET Public Objects/Interfaces Supported	396
SequeLinkCommand Object	397
SequeLinkCommandBuilder Object	398
SequeLinkConnection Object	399
SequeLinkDataAdapter Object	400
SequeLinkDataReader Object	401
SequeLinkError Object	401
SequeLinkErrorCollection Object	401
SequeLinkException Object	402
SequeLinkInfoMessageEventArgs Object	402
SequeLinkParameter Object	403
SequeLinkTrace Object	403
SequeLinkTransaction Object	405
Setting .NET Security Permissions	405
Code Access Permissions	405
Security Attributes	406
Error Handling	406
.NET Errors	406
ADO.NET Data Provider Errors	407
SequeLink® Server Errors	407
Database Errors	408
Diagnostic Support	408
Tracing Method Calls	408
PerfMon Support	410

- Designing .NET Applications for Performance 412
 - Selecting .NET Objects and Methods. 412
 - Designing .NET Applications 416
 - Retrieving Data. 423
 - Updating Data 426
- For More Information 427

Part 5: Reference

- A SQL Escape Sequences431**
 - Date, Time, and Timestamp Escape Sequences 432
 - Scalar Functions 432
 - String Functions 439
 - Numeric Functions 442
 - Date and Time Functions 444
 - System Functions 447
 - Like Predicate Escape Characters. 447
 - Outer Join Escape Sequences. 448
 - Procedure Call Escape Sequences 449
 - Procedure Call Escape Sequences 449
- B Data Types and Isolation Levels451**
 - Supported Data Types 451
 - DB2 UDB on z/OS 452
 - DB2 UDB on Linux, UNIX, and Windows 456
 - Informix. 462
 - Microsoft SQL Server 466
 - Oracle 473
 - Sybase 480
 - Isolation Levels 486
 - Using Snapshot Isolation Level (Microsoft SQL Server 2005 Only) 487

C	JDBC Support	489
	JDBC Compatibility	489
	Supported Functionality	489
D	JDBC Connection Pool Manager	541
	Creating a Data Source	541
	Creating a DataDirect SequeLink® Data Source Object	542
	Creating a Data Source Using the DataDirect®	
	Connection Pool Manager	544
	Connecting to a Data Source	547
	Closing the Connection Pool	550
E	Troubleshooting Using DataDirect Spy™	551
	Generating a DataDirect Spy™ Log	551
	Turning On and Off DataDirect Spy™ Logging	551
	ExtLogControl Class	552
	DataDirect Spy™ Log Example	553
F	Developing ODBC Applications for Internationalization	557
	Unicode and Non-Unicode ODBC Drivers	559
	Function Calls	559
	Data	563
	Developing ODBC Applications on Linux/UNIX	565
	Using Double-Byte Character Sets on Linux/UNIX	565
	Using UTF-16 for your Applications on Linux/UNIX	566
	The Driver Manager on Linux/UNIX	567

Values for IANAAppCodePage Connection String	
Attribute	569
Solaris	571
HP	572
AIX	573
Linux	574
Windows	576
G .NET Code Examples	577
Sample Tables Used in the Code Examples	577
Sample Tables for Oracle	578
Sample Tables for Sybase	579
Retrieving Data Using a DataReader	580
Using a Local Transaction With a DataReader	582
Using a Distributed Transaction	583
Using the CommandBuilder	586
Retrieving a Result Set Using a DataAdapter Object	587
Limiting the Rows Returned by a Select Statement	588
Updating Data in a DataSet	589
Calling a Stored Procedure	591
Retrieving Warning Information	593
Retrieving a Scalar Value	594
Index	595

List of Tables

Table 1-1.	ODBC Connection Attributes	61
Table 1-2.	Summary: Connection Failover Attributes for the ODBC Driver	79
Table 2-1.	Sources for Required ODBC Development Tools.	84
Table 2-2.	Compiler Requirements for Windows	84
Table 2-3.	Compiler Requirements for Linux and UNIX.	85
Table 2-4.	Function Conformance for 2.x ODBC Application	86
Table 2-5.	Function Conformance for 3.x ODBC Applications.	87
Table 2-6.	Support for Unicode ODBC W (Wide) Function Calls	90
Table 2-7.	Using SQLCancel in Multithreaded Applications	92
Table 2-8.	Support for Keyset-Driven Cursors (ODBC)	93
Table 3-1.	DataDirect Technologies Configuration Manager: Parts and Functions for ADO Data Sources	130
Table 3-2.	ADO Connection Attributes	157
Table 3-3.	Summary: Connection Failover Properties for the ADO Data Provider	167
Table 4-1.	Objects and Interfaces Supported by the ADO Data Provider	172
Table 4-2.	OLE DB Schema Rowsets Supported by the ADO Data Provider	174
Table 4-3.	OLE DB Property Groups Supported by the ADO Provider	175
Table 4-4.	OLE DB Data Source Property Supported by the ADO Data Provider	176
Table 4-5.	OLE DB Data Source Information Properties Supported by the ADO Data Provider	177
Table 4-6.	Initialization Properties Supported by the ADO Data Provider	182
Table 4-7.	Rowset Properties Supported by the ADO Data Provider	184

Table 4-8.	Session Properties Supported by the ADO Data Provider.	188
Table 4-9.	Supported OLE DB Interfaces Used by ADO.	188
Table 4-10.	Dynamic Properties Used for the ADO Command Object.	190
Table 4-11.	Mapping Methods Supported by the ADO Connection Object	193
Table 4-12.	Dynamic Properties Supported for the ADO Connection Object . . .	194
Table 4-13.	Mapping Methods Supported by the Recordset Object	201
Table 4-14.	Dynamic Properties Used for the Recordset Object.	203
Table 5-1.	JDBC Client Directory and Files.	220
Table 5-2.	Support for the Java Transaction API (JTA) by the JDBC Client	231
Table 5-3.	JDBC Connection Properties	237
Table 8-1.	Supported JDBC Features	328
Table 8-2.	Using Cancel in Multithreaded JDBC Applications	332
Table 8-3.	Support for Scroll-Sensitive Cursors (JDBC)	334
Table 9-1.	.NET Connection String Options.	370
Table 10-1.	Mapping of SequeLink Data Types	390
Table 10-2.	Mapping of the System.Data.DbTypes to SequeLinkDbTypes	392
Table 10-3.	Mapping .NET Framework Data Types to SequeLinkDbTypes	394
Table 10-4.	Properties of the SequeLinkCommand Object.	397
Table 10-5.	Properties of the SequeLinkCommandBuilder Object.	399
Table 10-6.	Properties of the SequeLinkConnection Object.	400
Table 10-7.	Properties of the SequeLinkParameter Object	403
Table 10-8.	Properties of the SequeLinkTrace Object.	404
Table 10-9.	Methods of the SequeLinkTransaction Object.	405
Table 10-10.	PerfMon Counters	411
Table A-1.	Language Features	431
Table A-2.	Scalar Functions Supported on DB2	433

Table A-3.	Scalar Functions Supported on Informix	435
Table A-4.	Scalar Functions Supported on Microsoft SQL Server.	436
Table A-5.	Scalar Functions Supported on Oracle.	437
Table A-6.	Scalar Functions Supported on Sybase	438
Table A-7.	Scalar String Functions	439
Table A-8.	Scalar Numeric Functions	442
Table A-9.	Scalar Time and Date Functions.	444
Table A-10.	Scalar System Functions	447
Table A-11.	Outer Join Escape Sequences Supported.	448
Table B-1.	Mapping Data Types for DB2 UDB on z/OS to ODBC Data Types. . . .	452
Table B-2.	Mapping Data Types for DB2 UDB for z/OS to ADO Data Types	453
Table B-3.	Mapping Data Types for DB2 UDB on z/OS to JDBC Data Types	454
Table B-4.	Mapping Data Types for DB2 UDB on z/OS to .NET Data Types. . . .	455
Table B-5.	Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ODBC Data Types	456
Table B-6.	Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ADO Data Types	457
Table B-7.	Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to JDBC Data Types	459
Table B-8.	Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to .NET Data Types	460
Table B-9.	Mapping Informix Data Types to ODBC Data Types	462
Table B-10.	Mapping the Informix Data Types to ADO Data Types.	463
Table B-11.	Mapping Informix Data Types to JDBC Data Types.	464
Table B-12.	Mapping Informix Data Types to .NET Data Types	465
Table B-13.	Mapping Data Types for Microsoft SQL Server to ODBC Data Types	466

Table B-14. Mapping Microsoft SQL Server Data Types to ADO Data Types 468

Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types 470

Table B-16. Mapping Microsoft SQL Server Data Types to .NET Framework
Types 471

Table B-17. Mapping the Oracle Data Types to ODBC Data Types 473

Table B-18. Mapping the Data Types for Oracle to ADO Data Types 475

Table B-19. Mapping the Data Types for Oracle to JDBC Data Types 477

Table B-20. Mapping Oracle Data Types to .NET Framework Types 478

Table B-21. Mapping the Data Types for Sybase to ODBC Data Types 480

Table B-22. Mapping the Data Types for Sybase to ADO Data Types 481

Table B-23. Mapping the Data Types for Sybase to JDBC Data Types 483

Table B-24. Mapping Data Types for Sybase Data Types to .NET Framework
Types 484

Table B-25. Isolation Levels 486

Table F-1. IANAAppCodePage Values 570

Table F-2. Code Pages Supported on Solaris 571

Table F-3. Code Pages Supported on HP-UX 572

Table F-4. Code Pages Supported on AIX 573

Table F-5. Code Pages Supported on Linux 574

Table F-6. Code Pages Supported on Windows 576

Preface

This book is your guide to developing client applications for DataDirect SequeLink® 6.0 from DataDirect Technologies. Read on to find out more about developing client applications to run in a SequeLink environment and how to use this book.

What Is DataDirect SequeLink®?

DataDirect SequeLink is a middleware product that provides point-to-point connections from a client to a server for the latest data access standards, including Open Database Connectivity (ODBC), JDBC, ActiveX Data Objects (ADO), and ADO.NET.

In this documentation, references to SequeLink Server and SequeLink Client apply to both the 32-bit and 64-bit versions. Information that applies to a specific version of SequeLink Server or SequeLink Client is identified.

Using This Book

This book assumes that you are familiar with your operating system and its commands; the definition of directories; and accessing a database through an end-user application.

This book contains the following information:

Part 1: Developing ODBC Applications

- [Chapter 1 “Using the ODBC Client” on page 31](#) provides information about using ODBC applications with the SequeLink Client *for* ODBC.
- [Chapter 2 “Developing ODBC Applications” on page 83](#) provides information about developing ODBC applications for SequeLink environments.

Part 2: Developing ADO Applications

- [Chapter 3 “Using the ADO Client” on page 127](#) provides information about using ADO applications with the SequeLink Client *for* ADO.
- [Chapter 4 “Developing ADO Applications” on page 171](#) provides information about developing ADO applications for SequeLink environments.

Part 3: Developing JDBC Applications

- [Chapter 5 “Using the JDBC Client” on page 215](#) provides information about using JDBC applications with the SequeLink Client *for* JDBC.
- [Chapter 6 “Using DataDirect Test™” on page 263](#) introduces DataDirect Test *for* JDBC, a development software component that allows you to test and learn the JDBC API. It also contains a tutorial that takes you through a working example of its use.
- [Chapter 7 “Tracking JDBC Calls” on page 315](#) introduces DataDirect Spy *for* JDBC, a development software component that allows you to track JDBC calls, and describes how to use it.
- [Chapter 8 “Developing JDBC Applications” on page 327](#) provides information about developing JDBC applications for SequeLink environments.

Part 4: Developing .NET Applications

- [Chapter 9 “Using the .NET Client” on page 363](#) provides information about using .NET applications with the SequeLink Client *for .NET*.
- [Chapter 10 “Developing .NET Applications” on page 389](#) provides information about developing .NET applications for SequeLink environments.

Part 5: Appendixes

- [Appendix A, “SQL Escape Sequences,” on page 431](#) describes the scalar functions supported for SequeLink. Your data store may not support all these functions.
- [Appendix B, “Data Types and Isolation Levels,” on page 451](#) lists the data types and isolation levels supported for each data store supported by SequeLink.
- [Appendix C, “JDBC Support,” on page 489](#) provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- [Appendix D, “JDBC Connection Pool Manager,” on page 541](#) provides sample code as an example of using the DataDirect Connection Pool Manager to allow your applications to handle connection pooling.
- [Appendix E, “Troubleshooting Using DataDirect Spy™,” on page 551](#) provides information to help you troubleshoot ODBC applications.
- [Appendix F, “Developing ODBC Applications for Internationalization,” on page 557](#) provides an overview of how to design your applications for internationalization, and provides the valid values for the IANAAppCodePage connection string attribute.
- [Appendix G, “.NET Code Examples,” on page 577](#) provides code examples of typical database access tasks in ADO.NET. All of the examples are written in C#.

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

Other SequeLink® Documentation

The following table provides a guide for finding information in your SequeLink documentation:

For information about...	Go to...
SequeLink concepts and planning your SequeLink environment	<i>Getting Started with SequeLink</i>
Installing the SequeLink middleware components	<i>SequeLink Installation Guide</i>
Administering your SequeLink environment	<i>SequeLink Administrator's Guide</i>
Developing ODBC, ADO, JDBC, and .NET applications for the SequeLink environment	<i>SequeLink Developer's Reference</i>
Troubleshooting and referencing error messages	<i>SequeLink Troubleshooting Guide and Reference</i>

HTML Version



All of these books can be placed on your system as HTML-based online help during a normal installation of the product. They are located in the help subdirectory of the product installation

directory. To use the help, you must have one of the following browsers installed:

- Internet Explorer 5.x or higher
- Netscape 4.x, 6.1, or higher
- FireFox 1.0 or higher

If you choose to install the online books, you can access the entire help system by selecting the help icon that appears in the DataDirect program group.



On UNIX and Linux platforms, if you want the help files, copy the /bookshtml subdirectory from the product DVD to a local directory.

To open the help system from a command-line environment, at a command prompt, enter:

```
browser_exe my_local_dir/bookshtml/help.htm
```

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

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

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

Help is available from the setup dialog box for the ODBC driver and ADO data provider. When you click **Help**, your browser opens to the correct topic in the help system, without opening

the help Table of Contents. A grey toolbar appears at the top of the browser window.



This tool bar contains previous and next navigation buttons.

PDF Version

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

http://www.datadirect.com/support/product_info/proddoc_product/index.ssp

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

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

- **To copy all product books**, copy the entire \bookspdf directory to your local or network drive.
- **To copy a specific book**, copy that book's directory structure (beneath the \bookspdf subdirectory) to your local or network drive. For example, to copy the *SequeLink Administrator's Guide*, you would copy the entire \admin subdirectory:

```
\bookspdf\admin
```

Maintaining the correct directory structure allows cross-book text searches and cross-references. If you download or copy the books

individually outside of their normal directory structure, their cross-book search indexes and hyperlinked cross-references to other volumes will not work. You can view a book individually, but it will not automatically open other books to which it has cross-references.

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

Typographical Conventions



This section uses the following typographical conventions:

Convention	Explanation
<i>italics</i>	Introduces new terms that you may not be familiar with, and is used occasionally for emphasis.
bold	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click Next .
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
<code>monospace</code>	Indicates syntax examples, values that you specify, or results that you receive.
<i>monospaced italics</i>	Indicates names that are placeholders for values you specify; for example, <i>filename</i> .
forward slash /	Separates menus and their associated commands. For example, Select File / Copy means to select Copy from the File menu.
vertical rule	Indicates an OR separator 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.
braces { }	Indicates that you must select one item. For example, <code>{yes no}</code> means you must specify either yes or no.
ellipsis . . .	Indicates that the immediately preceding item can be repeated any number of times in succession. An ellipsis following a closing bracket indicates that all information in that unit can be repeated.

Environment-Specific Information

This book supports users of various operating environments. Where it provides information that does not apply to all supported environments, the following symbols are used to identify that information.

Symbol	Environment
	<i>Windows.</i> Information specific to the Microsoft Windows 2000, Windows Server 2003, Windows XP, and Windows Vista environments is identified by the Windows symbol.
	<i>Linux and UNIX.</i> Information specific to Linux and UNIX 32-bit and 64-bit environments is identified by this symbol, which applies to all supported Linux and UNIX environments. UNIX is a registered trademark of The Open Group in the United States and other countries.
z/OS	<i>z/OS.</i> Information specific to z/OS environments is identified by the characters z/OS.

Contacting Technical Support

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

<http://support.datadirect.com>

The DataDirect Technologies Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

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

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

or contact your sales representative.

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

- The serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full customer information, including location.
- The DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your DataDirect product.

- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be recreated.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

Part 1: Developing ODBC Applications

This part contains the following chapters:

- [Chapter 1 “Using the ODBC Client” on page 31](#) provides information about using ODBC applications with the SequeLink Client *for* ODBC.
- [Chapter 2 “Developing ODBC Applications” on page 83](#) provides information about developing ODBC applications for SequeLink environments.

1 Using the ODBC Client

This chapter provides information about using ODBC applications with the SequeLink Client *for* ODBC (the ODBC Client).

About the ODBC Client

The ODBC Client supports ODBC applications through a component called the *SequeLink for ODBC driver*. On Linux, UNIX, and Windows platforms, the SequeLink *for* ODBC driver is compliant with the Microsoft Open Database Connectivity (ODBC) 3.5 specification.

ODBC is an Application Program Interface (API) specification that allows applications to access multiple database systems using Structured Query Language (SQL). ODBC provides maximum interoperability—a single application can access many different database systems. This allows 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 systems of their choice. The ODBC driver can connect all commercial ODBC-compliant applications with server databases.

For instructions on installing the ODBC Client, refer to the *SequeLink Installation Guide*.

Using the ODBC Administrator



The first step in setting up an ODBC connection is creating an ODBC data source. The ODBC Administrator is installed automatically when you install the ODBC Client on Windows. You use the ODBC Administrator to create and manage ODBC data sources.

To start the ODBC Administrator, click **Start**, then **Programs**. From the Programs menu, select **DataDirect SequeLink 6.0 Client for ODBC** or **DataDirect SequeLink 6.0 Client for ODBC 64-bit**, and then select the **ODBC Administrator** application. The ODBC Data Source Administrator window appears listing resident data sources.

NOTE: An ODBC Administrator does not exist for UNIX; you must edit the `odbc.ini` file using a text editor. See [“Configuring ODBC Client Data Sources on Linux and UNIX” on page 53](#) for instructions on creating ODBC client data sources for Linux and UNIX.

Configuring ODBC Client Data Sources on Windows

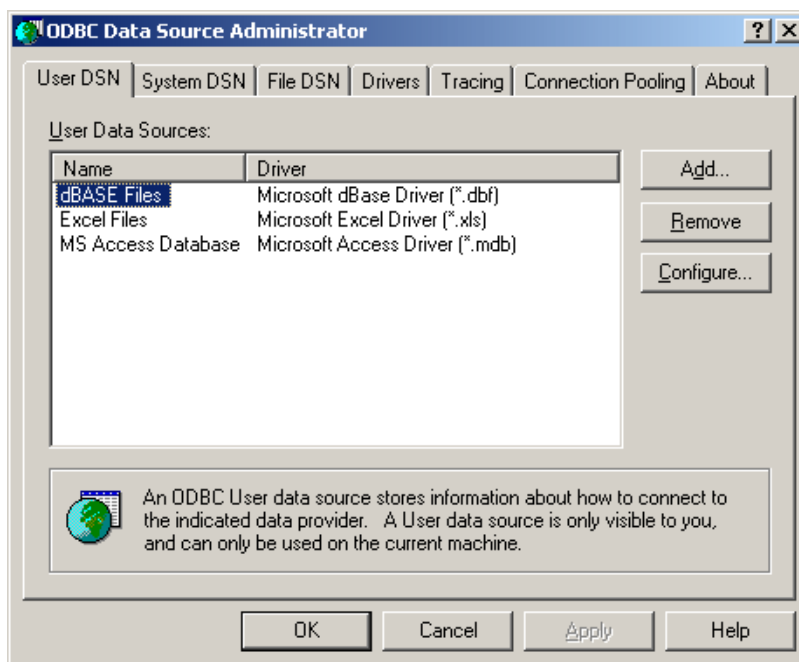


To configure 32-bit and 64-bit client data sources for the ODBC Client on Windows platforms, you use the ODBC Administrator.

Configuring ODBC User and System Client Data Sources

- 1 Start the ODBC Administrator. To start the ODBC Administrator, select **Start / Programs**. From the Programs menu, select **DataDirect SequeLink 6.0 Client for ODBC** or **DataDirect SequeLink 6.0 Client for ODBC 64-bit**, and then select the **ODBC Administrator** application.

- 2 Click the **User DSN** tab or the **System DSN** tab to list user or system data sources, respectively.



- 3 To configure a new data source, click the **Add** button. A list of installed drivers appears. Select **DataDirect SequeLink 6.0**; then, click **Finish**.

NOTE: To change an existing data source, select the data source you want to configure and click the **Configure** button.

The DataDirect SequeLink for ODBC Setup window appears.

The screenshot shows the 'DataDirect SequeLink for ODBC Setup' dialog box with the 'General' tab selected. The dialog has three tabs: 'General', 'Failover', and 'About'. The 'General' tab contains the following fields and controls:

- Data Source Name:** A text box containing 'Accounting'.
- Description:** An empty text box.
- Use LDAP:** An unchecked checkbox.
- SequeLink Server Host:** A text box containing 'localhost'.
- SequeLink Server Port:** A text box containing '16033'.
- Server Data Source:** A text box containing 'Default' and a browse button ('...').
- Distinguished Name:** An empty text box.
- Encrypted (SSL):** An unchecked checkbox.

Buttons on the right side of the dialog include 'Help' and 'Translate...'. At the bottom of the dialog are 'Test Connect', 'OK', 'Cancel', and 'Apply' buttons.

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

Data Source Name: Type a unique name that identifies this ODBC data source configuration. Examples are `Accounting` or `SequeLink to Oracle Data`.

Description: Optionally, type a description of the data source, for example, `My Accounting Database` or `Accounting Data in Oracle (SSL)`.

SequeLink Server Host: Type the TCP/IP host name of the SequeLink service to which the ODBC Client will connect.

SequeLink Server Port: Type the TCP/IP port the SequeLink service is listening on for connection requests. The port you specify must be the same port that was specified for the SequeLink service when the SequeLink Server was installed; the default is 19996.

Server Data Source: Type the name of a server data source configured for the SequeLink service to use for the connection, or click the ... button to select an existing server data source. This field is optional. If a server data source is not specified, the default server data source for that SequeLink service is used.

Use LDAP: To configure the ODBC Client to retrieve connection information from an LDAP directory, select the **Use LDAP** check box. The fields change on the lower half of the screen to accommodate the information required to query an LDAP server for connection information. Provide the following information:

LDAP Server Host: Type the TCP/IP host name of the LDAP server.

LDAP Server Port: Type the TCP/IP port the LDAP server is listening on for connection requests. The default value is 389.

Distinguished Name (DN): Type an identifier that uniquely identifies the LDAP entry where the connection information is stored.

Encrypted (SSL): If the remote SequeLink service is configured for Secure Sockets Layer (SSL) encryption, select this check box. If connecting to a SequeLink service enabled for SSL, you must select this check box.

When the check box is cleared (the default), communication between the SequeLink Client and SequeLink Server is not encrypted.

Configuration of encryption is performed on the SequeLink Server. For more information, refer to the *SequeLink Administrator's Guide*.

NOTES:

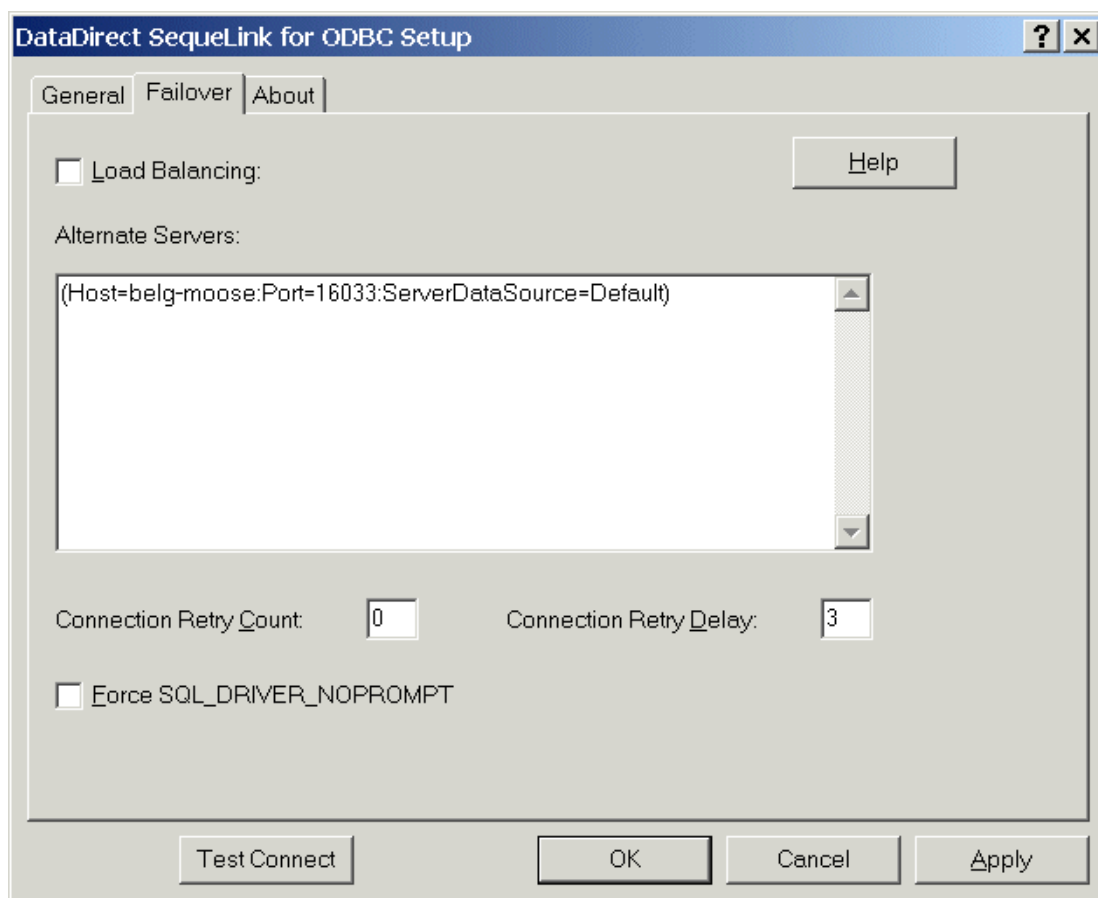
- An ODBC client data source can reference an LDAP directory to retrieve server connection information. For more information about retrieving connection information from LDAP directories, refer to the *SequeLink Administrator's Guide*.
- SSL encryption is not supported for LDAP Servers. The Use LDAP and the Encrypted (SSL) checkboxes are mutually exclusive.

Translate: Click **Translate** only if you want to configure an ODBC translator.

NOTE: We strongly recommend that you do not configure an ODBC translator; instead, rely on the native SequeLink transliteration between server and client code pages.

The Select Translator dialog box appears, listing translators specified in the ODBC Translators section of the system information file. Select a translator. When satisfied with your choice, click **OK** to close this dialog box and perform the translation.

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



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

Load Balancing: Select this check box to allow the driver to use client load balancing in its attempts to connect to primary and alternate database servers. In this case, the driver attempts to connect to the database servers in random order.

If this check box is not selected (the default), client load balancing is not used and the driver connects to each database server based on its sequential order (primary server first, then, alternate servers in the order they are specified).

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

The Load Balancing option is an optional setting that you can use in conjunction with connection failover. For a discussion of connection failover and for information about other connection options that you can set for this feature, refer to the *SequeLink Administrator's Guide*.

Alternate Servers: Type a list of alternate SequeLink servers to which the driver will try to connect if the primary SequeLink server is unavailable, using a string that defines the physical location of each alternate server. Specifying a value for this option enables connection failover for the driver.

IMPORTANT: If you specified an LDAP server in the LDAP Server Host field, the alternate servers *must* be LDAP servers.

The server name and port are required for each alternate server entry. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection. Currently, the only optional property that can be set for the alternate server is Server Data Source.

The string has the format:

```
(Host=servername1:Port=port1[:ServerDataSource=serverdatasourcename1], Host=servername2:Port=port2[:ServerDataSource=serverdatasourcename2], ...)
```

For example, the following Alternate Servers value defines two alternate SequeLink servers for connection failover:

```
(Host=server2:Port=19996:ServerDataSource=SDSN2,Host=server3:Port=19996:ServerDataSource=SDSN3)
```

If you are connecting to an LDAP server, the syntax includes the physical location of the server and the port number:

```
(Host=ld1.foo.com:Port=389,Host=ld2.foo.com:Port=
```

```
389,Host=ld3.foo.com:Port=389)
```

Connection Retry Count: Type a value to specify the number of times the driver tries to connect to the primary server and, if configured, to the alternate servers after the initial unsuccessful attempt.

Valid values are integers from 0 to 65535. When set to 0 (the default), the driver does not try to connect after the initial unsuccessful attempt.

If a connection is not established during the retry attempts, the driver returns an error that is generated by the first server to which it tried to connect.

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

For a discussion of connection failover and for information about other connection options that you can set for this feature, refer to the *SequeLink Administrator's Guide*.

Connection Retry Delay: Type a value to specify the number of seconds that the driver waits after the initial unsuccessful connection attempt before retrying a connection to the primary server and, if specified, to the alternate servers.

Valid values are integers from 0 to 65535. The default value is 3 (seconds). When set to 0, there is no delay between retries.

NOTE: This option has no effect unless the Connection Retry Count connection option is set to an integer value greater than 0.

This option and the Connection Retry Count connection option, which specifies the number of times the driver tries to connect after the initial unsuccessful attempt, are used in conjunction with connection failover.

For a discussion of connection failover and for information about other connection options that you can set for this feature, refer to the *SequeLink Administrator's Guide*.

Force SQL_DRIVER_NOPROMPT: Select this check box when connection failover or load balancing is enabled. This check box must be also selected if the application cannot change the DriverCompletion argument to SQL_DRIVER_NOPROMPT.

If this check box is not selected (the default), the behavior of the application is not changed.

- 6 At any point during the configuration process, you can click **Test Connect** to attempt to connect to the data source using the connection properties specified in the driver Setup dialog box. A logon dialog box appears; see ["ODBC Connection Attributes" on page 60](#) for details.

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

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

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

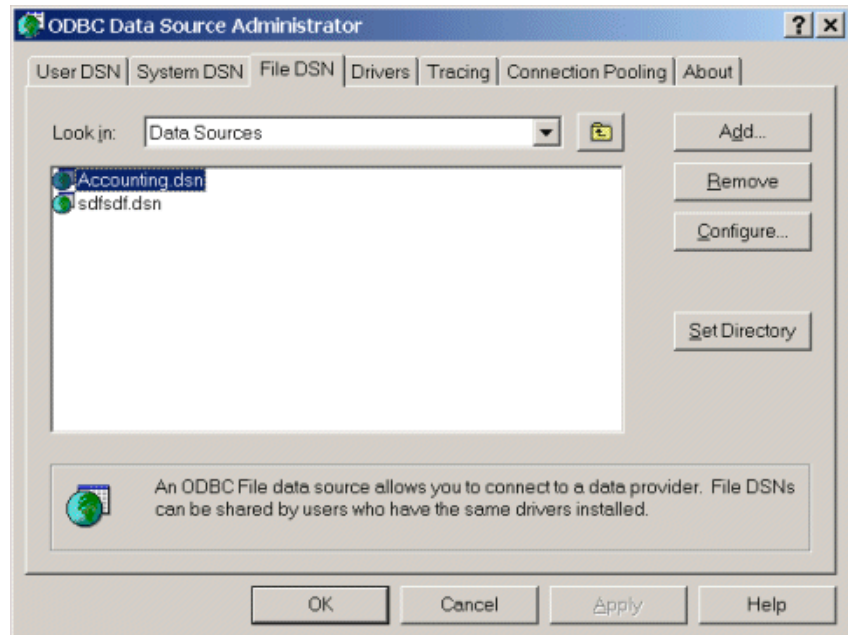
- 7 Click **OK** or **Cancel**. If you click **OK**, the values you have specified become the defaults when you connect to the data source. You can change these defaults by using this procedure to reconfigure your data source. You can override these defaults by connecting to the data source using a connection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

Configuring ODBC File Client Data Sources

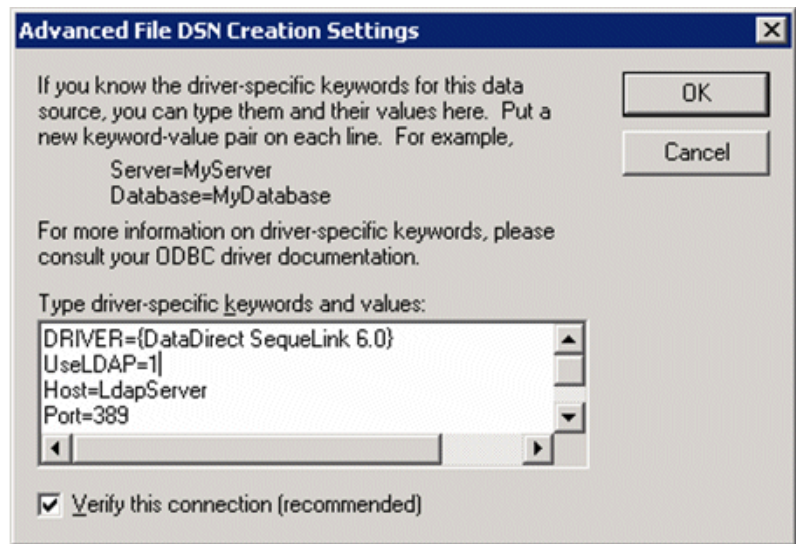
File data sources are data source files that can be stored on a file server, making the files available to any user who can access them. The 32-bit and 64-bit file-based data sources are created and used in the same manner.

To configure ODBC file client data sources:

- 1 Start the ODBC Administrator by clicking **Start**, then **Programs**. From the Programs menu, select **DataDirect SequeLink 6.0 Client for ODBC**, and then select the **ODBC Administrator** application.
- 2 Click the **File DSN** tab. The File DSN tab lists any file data sources in the specified directory.

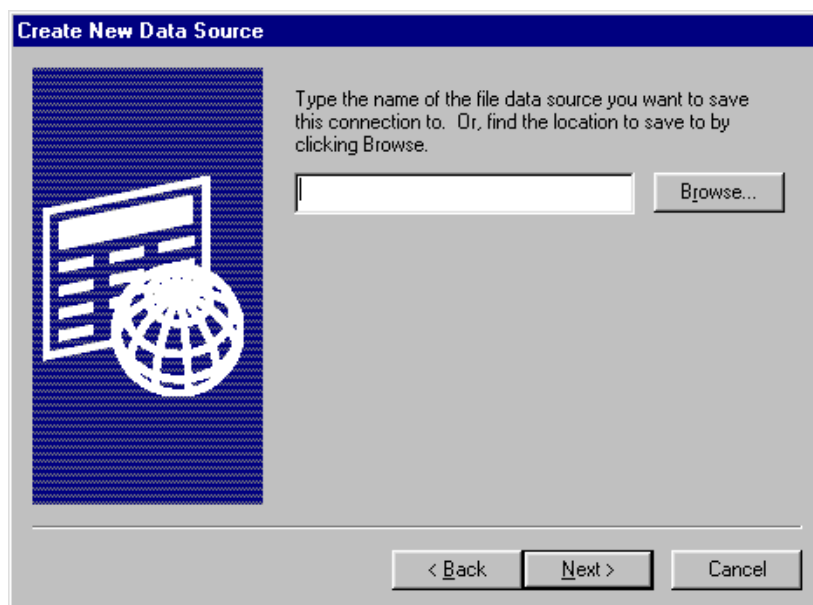


- 3 To configure a new data source, click the **Add** button. A list of installed drivers appears. Select **DataDirect SequeLink 6.0**; then, perform one of the following actions:
 - To configure the file data source to connect directly to a SequeLink Server without retrieving connection information from an LDAP directory, click **OK**. Then, skip to [Step 5](#).
 - To configure the file data source to retrieve connection information from an LDAP directory, continue with the next step.
- 4 Click **Advanced**. The Advanced File DSN Creation Settings window appears.



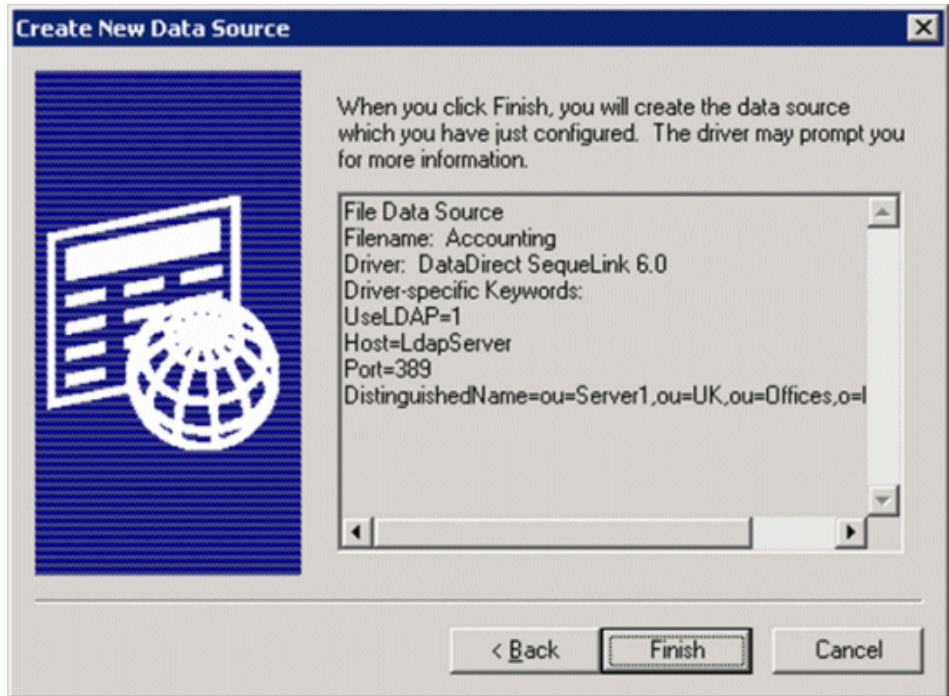
Type `UseLDAP=1` in the Type driver-specific keywords and values scrollable box; then, click **OK**. You are returned to the list of drivers. Click **Next** and continue with [Step 5](#).

5 The Create New Data Source window appears.



Type the name of the file data source you want to create or click **Browse** to select an existing file data source; then, click **Next**.

- 6 The Create New Data Source displays the settings you have configured for this data source.

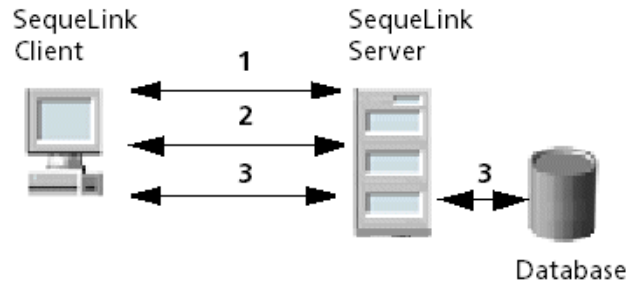


- 7 Click **Finish** to create the file data source.

A series of connection dialogs appear as described in "[ODBC Connection Dialogs](#)" on page 46. The file data source will be saved after you enter the correct information in the connection dialog boxes.

ODBC Connection Dialogs

A SequeLink data access connection involves the following stages:



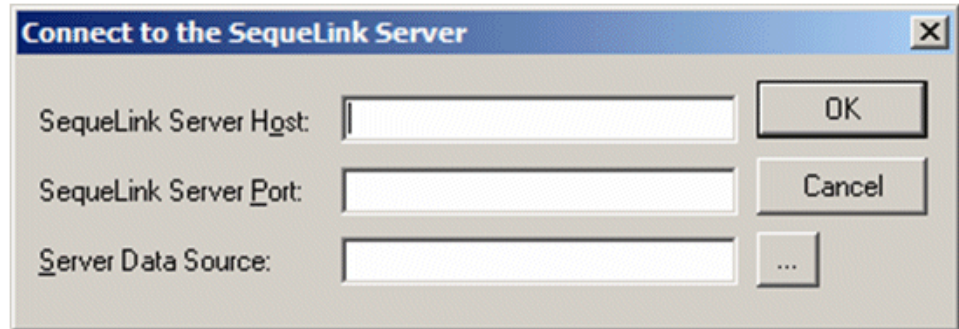
- 1 A network connection is established.
- 2 An authentication mechanism is used to establish the identity of the SequeLink Client to the SequeLink Server.
- 3 Based on information provided by the SequeLink Client application (for example, a database user name and password), a database connection is established.

Stage 1: Establishing a Network Connection

The first stage of the connection process involves establishing a network connection. The dialog box that appears depends on whether the connection has been configured to connect directly to a SequeLink service or to retrieve connection information for the SequeLink service from a centralized LDAP directory.

Connecting Directly to a SequeLink® Service

If the connection has been configured to connect directly to a SequeLink service, the Connect to the SequeLink Server dialog box appears.

The image shows a Windows-style dialog box titled "Connect to the SequeLink Server". It has a standard Windows title bar with a close button (X) in the top right corner. The dialog contains three text input fields stacked vertically. The first field is labeled "SequeLink Server Host:", the second is labeled "SequeLink Server Port:", and the third is labeled "Server Data Source:". To the right of these fields are three buttons: "OK" is positioned to the right of the first field, "Cancel" is to the right of the second field, and an ellipsis button ("...") is to the right of the third field. The dialog has a light gray background and a blue title bar.

Provide the following information; then, click **OK**.

SequeLink Server Host: Type the TCP/IP host name of the SequeLink service.

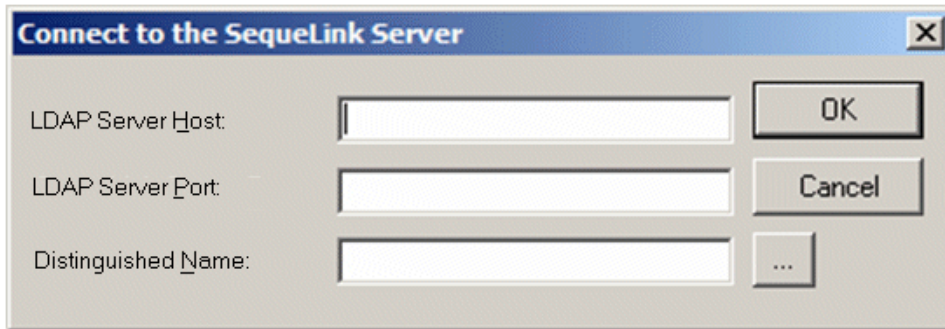
SequeLink Server Port: Type the TCP/IP port on which the SequeLink service is listening. A default installation of SequeLink Server uses the port 19996.

Server Data Source: Type the name of a server data source to use for the connection, or select one from the drop-down list. This step is optional. If a server data source is not specified, the default server data source for that service will be used for the connection.

Retrieving Connection Information from an LDAP Directory

If the connection has been configured to connect to an LDAP server to retrieve connection information from an LDAP directory, the Connect to the SequeLink Server dialog box appears.

For information about setting up an LDAP server for SequeLink, refer to the *SequeLink Administrator's Guide*.

A screenshot of a Windows-style dialog box titled "Connect to the SequeLink Server". The dialog has a blue title bar with a close button (X) in the top right corner. The main area is light gray and contains three text input fields. The first field is labeled "LDAP Server Host:", the second "LDAP Server Port:", and the third "Distinguished Name:". To the right of the first field is an "OK" button. To the right of the second field is a "Cancel" button. To the right of the third field is a button with three dots "...".

Provide the following information; then, click **OK**.

LDAP Server Host: Type the TCP/IP host name of the LDAP server.

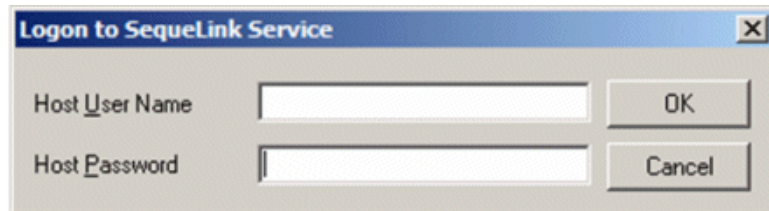
LDAP Server Port: Type the TCP/IP port on which the LDAP server is listening.

Distinguished Name: Type the Distinguished Name (DN) of the LDAP entry.

Stage 2: SequeLink® Server Authentication

The second stage of the connection process involves authentication of the SequeLink Client to the SequeLink Server. The dialog boxes that appear depend on how authentication is configured for the SequeLink service.

- When `ServiceAuthMethods=anonymous` or `ServiceAuthMethods=integrated_nt`, no dialog boxes appear.
- When `ServiceAuthMethods=OSLogon(HUID,HPWD)` or `ServiceAuthMethods=OSLogon(UID,PWD)`, the Logon to SequeLink Service dialog box appears.



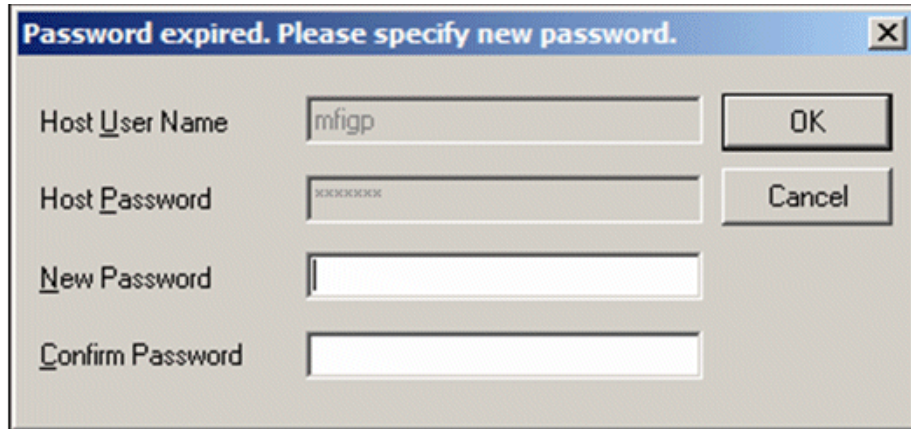
Provide the following information; then, click **OK**.

Host User Name: Type the host user name.

NOTE: When connecting to a Windows server, you must prefix the host user name with a server name, if authenticating to a local server, or a domain name (for example, SALES\DJONES). If the server name or domain name is omitted, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the machine on which the SequeLink Server is running. If this validation fails, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the domain of the machine on which the SequeLink Server is running.

Host Password: Type the host password.

- When `ServiceAuthMethods=OSLogon(HUID,HPWD,NPWD)` or `ServiceAuthMethods=OSLogon(UID,PWD,NPWD)` and the password is expired, the Password expired. Please specify new password dialog box appears.



NOTE: If the password is not expired, the Logon to SequeLink Service dialog box appears, prompting only for the host user name and host password.

Provide the following information; then, click **OK**.

Host User Name: Type the host user name.

NOTE: When connecting to a Windows server, you must prefix the host user name with a server name, if authenticating to a local server, or a domain name (for example, SALES\DJONES). If the server name or domain name is omitted, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the machine on which the SequeLink Server is running. If this validation fails, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the domain of the machine on which the SequeLink Server is running.

Host Password: Type the host password.

New Password: Type the new password to be used by the SequeLink password change mechanism.


Confirm Password: Type the new password again to confirm it.

For more information about configuring authentication, refer to the *SequeLink Administrator's Guide*.

Stage 3: Data Store Logon

The last stage of the connection process involves logging on the data store. The dialog boxes that appear depend on the data store logon method configured for the SequeLink service:

- When `DataSourceLogonMethod=OSIntegrated`, no dialog boxes appear.
- When `DataSourceLogonMethod=DBMSLogon(UID,PWD)` or `DataSourceLogonMethod=DBMSLogon(DBUID,DBPWD)`, a data store-specific user name and password are required and the Logon to SequeLink Service dialog box appears.

A screenshot of a Windows-style dialog box titled "Logon to SequeLink Service". The dialog has a blue title bar with a close button (X) in the top right corner. The main area is light gray and contains three text input fields. The first field is labeled "Database User Name:" and is followed by an "OK" button. The second field is labeled "Database Password:" and is followed by a "Cancel" button. The third field is labeled "Database:" and is at the bottom. All fields are currently empty.

Provide the following information; then, click **OK**.

Database User Name: Type the database logon ID.

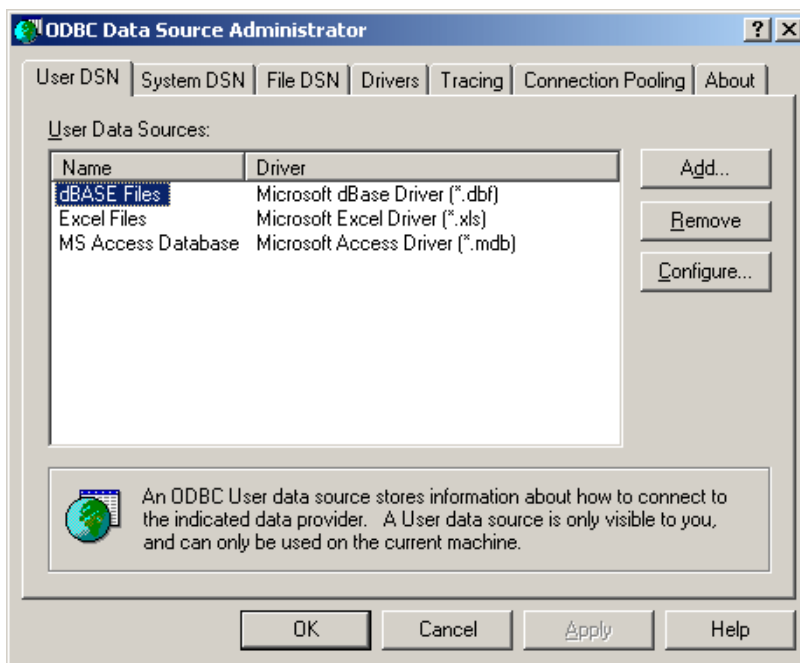
Database Password: Type the database password.

Database: Type the name of the database to which you want to connect. This field is disabled when the data store does not recognize the concept of databases.

For more information about configuring data store logon methods, refer to the *SequeLink Administrator's Guide*.

Testing ODBC Connections on Windows

- 1 On the SequeLink Client, start the ODBC Administrator. To start the ODBC Administrator, select **Start / Programs**. From the Programs menu, select **DataDirect SequeLink 6.0 Client for ODBC** or **DataDirect SequeLink 6.0 Client for ODBC 64-bit**, and then select the **ODBC Administrator** application. The ODBC Data Source Administrator window appears listing resident data sources.



- 2 Create an ODBC data source as described in ["Configuring ODBC User and System Client Data Sources"](#) on page 33,

specifying the TCP/IP address and TCP/IP port of the SequeLink service.

- 3 Click the **Test Connect** button to test the connection. If successful, a dialog appears telling you the connection was successful. You are now ready to start using your ODBC applications with SequeLink.

Configuring ODBC Client Data Sources on Linux and UNIX



For Linux and UNIX, an ODBC Administrator does not exist. This section describes how to configure the system information file and how to set some required environment variables to use the ODBC 32-bit Client and ODBC 64-bit Client on Linux and UNIX.

Configuring the System Information File

To configure an ODBC data source for Linux and UNIX, you must edit the `odbc.ini` file (32-bit Client) or `odbc64.ini` file (64-bit Client) using the attributes in [Table 1-1, "ODBC Connection Attributes," on page 61](#). In the following examples, notice that the description is the same for the 32-bit and 64-bit ODBC Client.

Example: odbc.ini for Solaris

The following code shows an example of an odbc.ini file for a 32-bit ODBC Client installed on a Solaris machine:

```
[ODBC Data Sources]
SALESDB=DataDirect SequeLink 6.0

[SALESDB]
Driver=path_of_installdir/lib/ivslk22.so
Description=DataDirect SequeLink 6.0
Host=
Port=
UseLDAP=0
DistinguishedName=
Encrypted=0
LoadBalancing=0
AlternateServers=
ConnectionRetryCount=0
ConnectionRetryDelay=3

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

where *path_of_installdir* is the path to the ODBC Client installation directory.

Example: odbc64.ini for Solaris

The following code shows an example of the odbc64.ini file for a 64-bit ODBC Client installed on a Solaris machine:

```
[ODBC Data Sources]
SALESDB=DataDirect SequeLink 6.0
```

```

[AccountingDB]
Driver=path_of_installdir/lib64/ivslk22.so
Description=DataDirect SequeLink 6.0
Host=
Port=
UseLDAP=0
DistinguishedName=
Encrypted=0
LoadBalancing=0
AlternateServers=
ConnectionRetryCount=0
ConnectionRetryDelay=3

[ODBC]
Trace=0
IANAAppCodePage=4
TraceFile=odbctrace.out
TraceDll=path_of_installdir/lib64/odbctrac.so
InstallDir=path_of_installdir

```

where *path_of_installdir* is the path to the ODBC Client installation directory.

Setting Environment Variables

You must set several environment variables for the ODBC Client on Linux and UNIX by executing a shell script located in the installation directory.

To execute the shell script:

- If you are using the Bourne or Korn shell, type:
 - `. sqlnk.sh` (32-bit client)
 - `. sqlnk64.sh` (64-bit client)
- If you are using the C shell, type:
 - `source sqlnk.csh` (32-bit client)
 - `source sqlnk64.csh` (64-bit client)

Executing this shell script sets the following environment variables:

ODBCINI	Specifies where the centralized <code>odbc.ini</code> or <code>odbc64.ini</code> file is located.
SQLNK_ODBC_HOME	Specifies the full path of the directory containing the ODBC Client shared libraries.

Executing this shell script also sets the appropriate library search environment variable (`LD_LIBRARY_PATH` on Solaris and Linux, `SHLIB_PATH` on HP-UX, or `LIBPATH` on AIX).

Using a Centralized System Information File

Because Linux and UNIX are multi-user environments, you may want to use a single centralized `odbc.ini` file controlled by a system administrator. To do this, set the `ODBCINI` environment variable to point to the fully qualified pathname of the centralized file.

For example:

- In the Bourne or Korn shell, type:

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

- In the C shell, type:

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

The `odbc.ini` file also require an `[ODBC]` section that includes the `InstallDir` keyword. The value of the `InstallDir` keyword must be the path to the directory that contains the `/lib` and `/messages` directories.

For example, if you choose the default installation directory for the 32-bit ODBC Client, the following line must be in the [ODBC] section of the `odbc.ini` file:

```
InstallDir=/usr/slodbc60
```

Connecting Using a Connection String

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

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 data source 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 data source.

The DSN connection string has the form::

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

For example, a connection string for SequeLink may look like this:

```
DSN=Accounting;DB=EMP;UID=JOHN;PWD=XYZZY
```

or

```
DSN=Accounting;DB="X:IV;EMP";UID=JOHN;PWD=XYZZY
```

NOTE: If the database name (DB) contains a semicolon (;), you must place the name in quotes, as shown in the preceding example.

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

The DSN-less connection string has the form:

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

NOTE: Empty string is the default value for attributes that use a string value unless otherwise noted.

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

```
DRIVER=DataDirect SequeLink 6.0;DB=Emp;UID=JOHN;PWD=XYZZY
```

See [“ODBC Connection Attributes” on page 60](#) for a list of ODBC connection attributes and their valid values.

See [“DSN-less Connections in Linux and UNIX” on page 58](#) for more information about using DSN-less connections.

DSN-less Connections in Linux and UNIX



Connections to a data source can be made via a connection string without referring to a data source name (DSN-less connections). This is accomplished by specifying the "DRIVER=" instead of the "DSN=" keyword in a connection string, as outlined in the ODBC specification. For this to work on Linux and UNIX, a file called `odbcinst.ini` must exist when the driver encounters `DRIVER=` in a connection string.

By default, Setup installs a sample `odbcinst.ini` file in the same location as the sample `odbc.ini` file, which is in the product installation directory. See [“Configuring the System Information File” on page 53](#) for an explanation of the `odbc.ini` file. The environment variable `ODBCINST`, recognized by the DataDirect SequeLink *for* ODBC driver, must be set to point to the fully qualified path name of the `odbcinst.ini` file.

For example, to point to the default location of the file in the C shell, you set this variable as follows:

```
setenv ODBCINST /opt/sl60/client/odbcinst.ini
```

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

```
ODBCINST=/opt/sl60/client/odbcinst.ini;export ODBCINST
```

If the `ODBCINST` variable is not set, the driver looks in the user’s home directory for a file named `odbcinst.ini`. If the driver does not find the file, it returns the message:

```
HY000 - "ODBCINST.INI is not available in the directory
pointed to by the ODBCINST environment variable (or the
current user's HOME directory) and therefore DSN-Less
connections cannot be made."
```

The following is a sample `odbcinst.ini`.

```
[ODBC Drivers]
DataDirect SequeLink 6.0 = Installed

[DataDirect SequeLink 6.0]
APILevel=1
ConnectFunctions=Y
Driver=ivslk22.so
DriverODBCVer=3.52
FileUsage=0
SQLLevel=1
```

ODBC Connection Attributes

Table 1-1 lists ODBC connection attributes in alphabetical order. The list includes long and short names and provides a description of each attribute. Short names are shown enclosed within parentheses ().

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

Table 1-1. ODBC Connection Attributes

Attribute	Description
AlternateServers (ASRV)	<p>A list of alternate SequeLink servers to which the driver will try to connect if the primary database server is unavailable. Specifying a value for this attribute enables connection failover for the driver.</p> <p>The value must be in the form of a string that defines the physical location of each alternate server. The Host and Port values are required for each alternate server entry. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection. Currently, the only optional connection attribute that can be set for the alternate server is ServerDataSource.</p> <p>The string has the format:</p> <pre>(Host=servername1:Port=port1 [:ServerDataSource=datasourcename],...)</pre> <p>For example, the following AlternateServers value defines three alternate SequeLink servers for connection failover:</p> <pre>AlternateServers=(Host=AccountingSLServer: Port=13999,Host=AccountingAltServer:Port= 13998:ServerDataSource=Backup,Host= AccountingAlt2:Port=13997)</pre> <p>IMPORTANT: If you specify an LDAP server in the Host attribute, the alternate servers must be LDAP servers. For example, the following AlternateServers value defines three alternate LDAP servers for connection failover:</p> <pre>AlternateServers=(Host=ld1.foo.com:Port= 389,Host=ld2.foo.com:Port=389,Host= ld3.foo.com:Port=389)</pre> <p>See “Configuring Connection Failover” on page 76 for a discussion of connection failover.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
ApplicationID (APPID)	<p>Specifies the application ID that identifies the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>See “Specifying Application IDs” on page 99 for more information about using application IDs to limit access to the SequeLink services.</p>
ApplicationName (APPNAME)	<p>Identifies the application that is establishing the connections.</p> <p>The initial default is SequeLink for ODBC Application.</p>
AutomaticApplicationID (AUTOAPPID)	<p>Specifies an application ID that is automatically generated by the ODBC Client to identify the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>See “Specifying Application IDs” on page 99 for more information about using application IDs to limit access to SequeLink services.</p>
BlockFetchForUpdate (BFFU)	<p>BlockFetchForUpdate={0 1}. Specifies a workaround connection attribute. When the isolation level is Read committed and a SELECT FOR UPDATE statement is issued against some data stores, the ODBC Client does not lock the expected row.</p> <p>When set to 0, the appropriate row is locked.</p> <p>When set to 1 (the initial default), the appropriate row is not locked.</p> <p>NOTE: Specifying 0 will degrade the performance for SELECT FOR UPDATE statements because rows will be fetched one at a time.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
ConnectionRetryCount (CRC)	<p>Specifies the number of times the driver tries to connect to the primary server, and if configured, to the alternate servers after the initial unsuccessful attempt.</p> <p>Valid values are integers from 0 to 65535. When set to 0 (the initial default), the driver does not try to connect after the initial unsuccessful attempt.</p> <p>If a connection is not established during the retry attempts, the driver returns an error that is generated by the last server to which it tried to connect.</p> <p>This attribute and the ConnectionRetryDelay connection string attribute, which specifies the wait interval between attempts, can be used in conjunction with connection failover.</p> <p>See “Configuring Connection Failover” on page 76 for a discussion of connection failover and for information about other connection string attributes that you can set for this feature.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
ConnectionRetryDelay (CRD)	<p>Specifies the number of seconds that the driver waits after the initial unsuccessful connection attempt before retrying a connection to the primary server and, if specified, to the alternate servers.</p> <p>Valid values are integers from 0 to 65535. The initial default is 3 (seconds). When set to 0, there is no delay between retries.</p> <p>NOTE: This attribute has no effect unless the ConnectionRetryCount connection string attribute is set to an integer value greater than 0.</p> <p>This attribute and the ConnectionRetryCount connection string attribute, which specifies the number of times the driver tries to connect after the initial unsuccessful attempt, can be used in conjunction with connection failover.</p> <p>See “Configuring Connection Failover” on page 76 for a discussion of connection failover and for information about other connection string attributes that you can set for this feature.</p>
Database (DB)	<p>Specifies the name of the database to which you want to connect.</p>
DBLogonID (DBUID)	<p>Specifies the data store user name, which may be required depending on the server configuration.</p>
DBPassword (DBPWD)	<p>Specifies the data store password, which may be required depending on the server configuration.</p>
DistinguishedName (DN)	<p>Specifies the distinguished name identifying the LDAP entry from which connection information is retrieved. This attribute is required when UseLDAP=1.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
DriverCompletion (DCOMP)	<p>DriverCompletion={0 1}. Determines whether to overwrite the DriverCompletion argument setting defined by the application. This attribute must be enabled when connection failover or load balancing is set up.</p> <p>When set to 1, the driver overwrites the DriverCompletion parameter of the SQLDriverConnect call with the value SQL_DRIVER_NOPROMPT.</p> <p>When set to 0 (the default), the behavior of the application is not changed.</p>
DSN (DSN)	<p>Specifies a string that identifies an ODBC data source configuration. Examples include Accounting or SequeLink to Oracle Data.</p>
EnableDescribeParam (EDP)	<p>EnableDescribeParam={0 1}. Specifies a workaround connection attribute for connections to Oracle data stores only.</p> <p>When set to 0 (the initial default), support is turned off for SQLDescribeParam.</p> <p>When set to 1, support is turned on for SQLDescribeParam and will describe all parameters as SQL_CHAR with a precision of 999.</p>
Encrypted (ENC)	<p>Encrypted={0 1}. Enables the use of SSL encryption IF the remote SequeLink Service the client is connecting to is configured for SSL.</p> <p>When set to 0 (the default), the driver does not use SSL encryption for data exchanged with the SequeLink Server.</p> <p>When set to 1, the driver uses SSL encryption. This attribute must be set to 1 when connecting to a SequeLink service enabled for SSL.</p> <p>NOTE: The Encrypted connection attribute is mutually exclusive with the UseLDAP connection attribute.</p>

Table 1-1. ODBC Connection Attributes *(cont.)*

Attribute	Description
FetchNextOnly (FNO)	<p>FetchNextOnly={TRUE FALSE}. Turns on a workaround for Visual Basic/Remote Data Objects (RDO) that circumvents a problem with FORWARD_ONLY cursors when the driver reports other values than FETCH_NEXT for SQLGetInfo(SQL_FETCH_DIRECTION).</p> <p>For example, if the driver only reports FETCH_NEXT, RDO performs SQLExecDirect, SQLBindCol, and SQLExtendedFetch(NEXT). If the driver supports more than FETCH_NEXT, RDO performs SQLExecDirect, SQLExtendedFetch(NEXT), and SQLGetData. This is only valid when the rowsize is 1, but RDO uses a larger rowsize in this situation.</p> <p>When set to TRUE, the driver will incorrectly report that only SQL_FETCH_NEXT is supported, which satisfies RDO.</p> <p>When set to FALSE (the initial default), the driver will correctly report other values than SQL_FETCH_NEXT.</p>
FixCharTrim (FCT)	<p>FixCharTrim={0 1}. Turns on a workaround for applications that have a problem using SQL_CHAR data padded with spaces. The ODBC driver returns SQL_CHAR data padded with spaces as mandated by the ODBC specification.</p> <p>When set to 0 (the initial default), the workaround is turned off.</p> <p>When set to 1, SQL_CHAR data that is not padded with spaces is returned.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
GetOutputParams (GOP)	<p>Turns on a workaround that allows you to control when output parameters of stored procedures are returned to calling applications. This attribute uses a bitmask with the following options:</p> <p>When set to 1, output parameters are returned after a SQLExecute.</p> <p>When set to 2, output parameters are returned after a SQLFetch is complete.</p> <p>When set to 4, output parameters are returned after SQLMoreResults returns no more rows.</p> <p>When set to 7 (the initial default), output parameters are returned after all of the above.</p> <p>The value for this connection attribute should be set to the cumulative value of all chosen options added together.</p> <p>NOTE: Set GetOutputParams=3 when executing stored procedures with output parameters in RDO (Visual Basic 5 and 6).</p>
HLogonID (HUID)	Specifies the host user name, which may be required depending on the server configuration.
HPassword (HPWD)	Specifies the host password, which may be required depending on the server configuration.
Host (HST)	<p>Specifies the TCP/IP address of the SequeLink Server, specified in dotted format or as a host name.</p> <p>LDAP: If LDAP is enabled, this attribute identifies the TCP/IP address of the LDAP server. This attribute can also be a list of LDAP servers separated by a blank space (for example, ld1.foo.com ld2.foo.com ld3.foo.com). If the first LDAP server in the list does not respond, the ODBC Client will try to connect to the next LDAP server in the list.</p>

Table 1-1. ODBC Connection Attributes *(cont.)*

Attribute	Description
IANAAppCodePage (IACP)	<p>Valid values for this attribute are listed in “Values for IANAAppCodePage Connection String Attribute” on page 569. 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 IANAAppCodePage value in the following order:</p> <ul style="list-style-type: none">■ In the connection string■ In the DataSource section of the system file (odbc.ini)■ In the ODBC section of the system file (odbc.ini)■ If no IANAAppCodePage value is found, the driver uses the default value of 4 (ISO 8859-1 Latin-1).

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
LimitCursorColumnsize (LCCS)	<p>Specifies the Columnsize of a cursor variable. For example, when a default varchar column is created in FileMaker, the database defines it at 1000000 chars. When the application requests a static cursor on a Unicode data source, the driver allocates 3000000 (UTF-8) bytes for this type of column. When multiple varchar columns exist, the application slows down because of disk access in the static cursor. In practice, no applications use 1000000 chars; thus, limiting the column size makes the rowbuffer size smaller and reduces disk access. The ODBC application can specify a parameter in the connection string, for example, LimitCursorColumnsize=1000.</p> <p>When set to 0 (the default), the workaround is not enabled.</p> <p>NOTE: On the SequeLink Server, this workaround can be set using the attribute DataSourceLimitCursorColumnsize. See the <i>SequeLink Administrator's Guide</i> for information about this service attribute.</p>
LimitParameterBindSize (LPBS)	<p>Specifies the parameter bind size for SQL_CHAR, SQL_VARCHAR, SQL_BINARY, and SQL_VARBINARY values. When exporting a table from Microsoft Access that contains Null data in a memo column, Microsoft Access binds the parameter with a columnSize of 4294967295. The SequeLink Client attempts to allocate a buffer of this size, which typically fails due to lack of memory.</p> <p>To get around this application bug, the SequeLink Server can be configured to limit the parameter bind size for SQL_CHAR, SQL_VARCHAR, SQL_BINARY, and SQL_VARBINARY values to a reasonable value.</p>

Table 1-1. ODBC Connection Attributes *(cont.)*

Attribute	Description
LoadBalancing (LB)	<p>LoadBalancing={0 1}. Determines whether the driver uses client load balancing in its attempts to connect to primary and alternate database servers.</p> <p>When set to 1, the driver attempts to connect to the database servers in random order.</p> <p>When set to 0 (the initial default), client load balancing is not used and the driver connects to each database server based on its sequential order (primary server first, then, alternate servers in the order they are specified).</p> <p>NOTE: This attribute has no effect unless alternate servers are defined for the AlternateServers connection string attribute.</p> <p>The LoadBalancing connection string attribute is an optional setting that you can use in conjunction with connection failover.</p> <p>See “Using Client Load Balancing” on page 81 for more information and for information about other connection string attributes that you can set for this feature.</p>
LogonID (UID)	<p>Specifies the host or data store user name, which may be required depending on the server configuration.</p>
MSAccessWorkaroundCreateParams (MAWCP)	<p>MSAccessWorkaroundCreateParams={0 1}. Specifies a workaround for a bug in Microsoft Access that generates erroneous queries for Table Export when the value of a CREATE_PARAMS column of SQLGetTypeInfo contains a precision for TIMESTAMP.</p> <p>When set to 1, the workaround is enabled.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
NewPassword (NPWD)	<p>Specifies the new host password to be used. If specified and applicable to the connection, the SequeLink password change mechanism is invoked. When the password has been changed successfully, the following warning is generated:</p> <pre>[DataDirect][ODBC SequeLink driver][SequeLink Server] The user password was changed successfully</pre> <p>If unspecified and the SequeLink Server detects that the host password has expired, you will be prompted for a new host password.</p> <p>For more information about the SequeLink password change mechanism, refer to the <i>SequeLink Administrator's Guide</i>.</p>
Password (PWD)	Specifies the host or data store password, which may be required depending on the server configuration.
Port (PRT)	<p>Specifies the TCP/IP port on which the SequeLink Server is listening.</p> <p>LDAP: If LDAP is enabled, this attribute identifies the TCP/IP port on which the LDAP server is listening. If you do not specify a port, the default port for LDAP (389) will be used.</p>
ServerDataSource (SDSN)	Optionally, specifies a string that identifies the server data source to be used for the connection. If not specified, the configuration of the default server data source will be used for the connection.
SessionConnectTimeout (SCTO)	<p>Imposes a time limit on:</p> <ul style="list-style-type: none"> ■ The establishment of the TCP/IP connection with the server. ■ The establishment of the SequeLink session via an initial handshake with the server (this includes the time to initialize the necessary server processes and/or threads).

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
SLKStaticCursorLongColBuffLen (SSCLCBL)	<p>Turns on a workaround that allows you to specify the amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with a static cursor.</p> <p>Because the driver caches Unicode characters (UTF-16-LE on Windows, UTF-8 on UNIX), the number of characters that can be cached is smaller for the long-character, Unicode columns.</p> <p>The initial default is 4.</p>
TransliterationWorkAroundServer (TWAS)	<p>Turns on a workaround for multiple transliteration workarounds. Refer to the <i>SequeLink Administrator's Guide</i> for more information about how SequeLink handles transliteration.</p> <p>When set to 1 or 2, this workaround resolves transliteration issues between Shift-JIS/Windows-31j and eucJP by mapping "look-alike" characters.</p> <p>When set to 0 (the initial default), the workaround is not enabled.</p>
UseLDAP (LDAP)	<p>UseLDAP={0 1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP.</p> <p>When set to 0 (the initial default), the SequeLink Client will connect directly to the specified SequeLink Server.</p> <p>When set to 1, the SequeLink Client will retrieve the TCP/IP host, TCP/IP port, and SequeLink server data source (optional) from an LDAP entry identified by a Distinguished Name (DN). Once the connection information is retrieved, the SequeLink Client will connect directly to the specified SequeLink Server. The DistinguishedName (DN) attribute is required.</p> <p>NOTE: The Encrypted connection attribute is mutually exclusive with the UseLDAP connection attribute.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
WorkArounds (WA)	<p>Turns on workarounds that allow you to take full advantage of the ODBC driver with ODBC applications requiring nonstandard or extended behavior.</p> <p>IMPORTANT: Each of these options has potential side effects related to its use. An option should only be used to address the specific problem for which it was designed.</p> <p>When set to 1, the ODBC driver returns 1, allowing Microsoft Access to open tables as read-write. If an ODBC driver reports to Microsoft Access 2.0 that its SQL_CURSOR_COMMIT_BEHAVIOR or SQL_CURSOR_ROLLBACK_BEHAVIOR is 0, Microsoft Access opens tables as read-only.</p> <p>When set to 2, the driver reports that qualifiers are not supported. This option is provided because some applications cannot handle database qualifiers.</p> <p>When set to 4, the driver detects when Visual Basic requires multiple connections to a DBMS and has the multiple ODBC connections share a single physical connection to the DBMS. For DBMSs that support only a single connection, the second attempt fails.</p> <p>When set to 8, the driver returns 1. However, if an ODBC driver cannot detect the number of rows that are affected by an Insert, Update, or Delete statement, it may return -1 in SQLRowCount. Some products cannot handle this.</p> <p>When set to 16, the driver returns no INDEX_QUALIFIER, allowing Microsoft Access to open the table. If an ODBC driver in SQLStatistics reports to Microsoft Access 1.1 that an INDEX_QUALIFIER contains a period, Microsoft Access returns a <code>tablename is not a valid name</code> error.</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
WorkArounds (cont.)	<p>When set to 32, users of flat-file drivers are allowed to abort a long-running query by pressing the ESC key.</p> <p>When set to 64, the result is a column name of <i>Cposition</i> where <i>position</i> is the ordinal position in the result set. For example:</p> <pre>SELECT col1, col2+col3 FROM table1</pre> <p>produces the column names col1 and C2. SQLColAttributes/SQL_COLUMN_NAME returns an empty string for result columns that are expressions. Use this option for applications that cannot handle empty strings in column names.</p> <p>When set to 256, SQLGetInfo/SQL_ACTIVE_CONNECTIONS is forced to return as 1.</p> <p>When set to 512, the SQLSpecialColumns function returns a unique index as returned from SQLStatistics to prevent ROWID results.</p> <p>When set to 2048, SQLDriverConnect returns Database= instead of DB= in the returned connection string.</p> <p>When set to 65536, trailing zeros are stripped from decimal results, which prevents Microsoft Access from generating an error when decimal columns containing trailing zeros are included in the unique index.</p> <p>When set to 131072, all occurrences of the double quote character (" ") are turned into the accent grave character (`). Some applications always quote identifiers with double quotes. Double quoting causes problems for data sources that do not return SQLGetInfo/SQL_IDENTIFIER_QUOTE_CHAR = " .</p>

Table 1-1. ODBC Connection Attributes (cont.)

Attribute	Description
WorkArounds (cont.)	<p>When set to 524288, the precision and scale settings for SQL_DECIMAL parameters are overridden to precision 40 and scale 20.</p> <p>When set to 8388608, SQLGetInfo/SQL_DATABASE_NAME is returned as an empty string when SQLGetInfo/SQL_MAX_QUALIFIER_NAME_LEN is 0. This option should be used with Inprise/Borland tools, such as Delphi.</p> <p>When set to 536870912, SQLBindParameter is allowed to be called after SQLExecute to change the precision of previously bound parameters.</p> <p>When set to 1073741824, Microsoft Access assumes that ORDER BY columns do not have to be in the SELECT list. This option provides a workaround for data stores that always use ORDER BY columns.</p>
WorkArounds2 (WA2)	<p>Turns on workarounds that allow you to take full advantage of the ODBC driver with ODBC applications requiring nonstandard or extended behavior.</p> <p>IMPORTANT: Each of these options has potential side effects related to its use. An option should only be used to address the specific problem for which it was designed.</p> <p>When set to 2, the driver ignores the ColumnSize/DecimalDigits specified by the application and uses the database defaults instead. Some applications incorrectly specify ColumnSize/DecimalDigits when binding timestamp parameters.</p>

Table 1-1. ODBC Connection Attributes *(cont.)*

Attribute	Description
WorkArounds2 (WA2) <i>(cont.)</i>	<p>When set to 4, Microsoft Access uses the most recent native type mapping, as returned by SQLGetTypeInfo, for a specific SQL type. This option reverses the order in which types are returned, so that Microsoft Access will use the most appropriate native type. This option is recommended if you are using Microsoft Access against an Oracle8.x data store.</p> <p>When set to 32, Microsoft Access requires that the characters "DSN=" are returned by SQLDriverConnect in the connection string output parameter.</p>

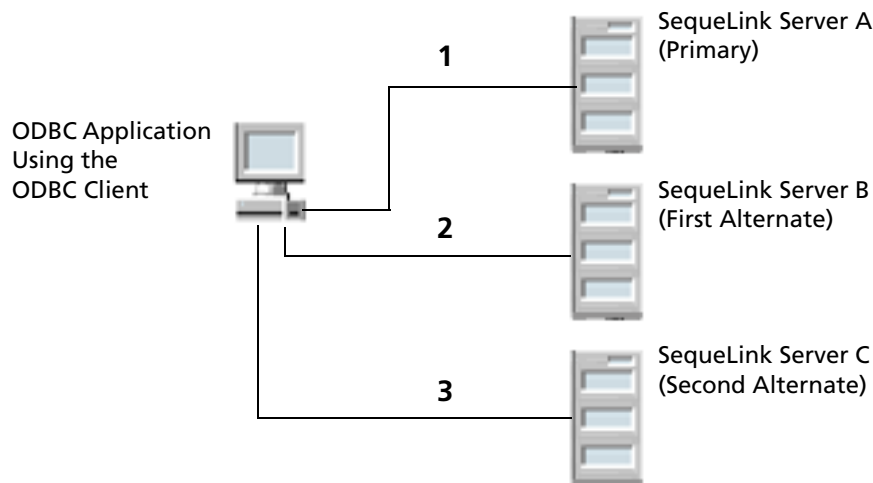
Configuring Connection Failover

Connection failover allows an application to connect to an alternate, or backup, SequeLink server if the primary database server is unavailable, for example, because of a hardware failure or traffic overload. Connection failover ensures that the data on which your critical ODBC applications depend is always available.

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

For example, suppose you have the environment shown in [Figure 1-1](#) with multiple SequeLink servers: SequeLink Server A, B, and C. SequeLink Server A is designated as the primary database server, SequeLink Server B is the first alternate server, and SequeLink Server C is the second alternate server.

Figure 1-1. Configuring Connection Failover with the ODBC Client



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

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the driver can retry each alternate SequeLink Server (primary and alternate) for a specified number of attempts. You can specify the number of attempts that are made through the connection retry feature. You can also specify the number of seconds of delay, if any, between attempts through the connection delay feature. See ["Using Connection Retry" on page 82](#) for more information about connection retry.

The ODBC Client fails over to the next alternate SequeLink server only if a successful connection cannot be established with the

current alternate server. If the ODBC Client successfully establishes communication with a SequeLink Server and the connection request is rejected by the SequeLink Server because, for example, the login information is invalid, then the driver generates an error and does not try to connect to the next SequeLink Server in the list.

Connection failover provides protection for new connections only and does not preserve states for transactions or queries. For details on configuring connection failover, refer to the *SequeLink Administrator's Guide*.

To configure connection failover, you **must** specify a list of alternate SequeLink Servers that are tried at connection time if the primary server is not accepting connections. To do this, use the AlternateServers connection option.

Optionally, you can specify the following additional connection failover features:

- The number of times the ODBC Client attempts to connect to the primary and alternate SequeLink servers after the initial unsuccessful connection attempt. By default, the ODBC Client does not retry. To set this feature, use the Connection Retry Count (ConnectionRetryCount) connection option. See ["Using Connection Retry" on page 82](#) for more information.
- The wait interval, in seconds, between attempts to connect to the primary and alternate SequeLink servers. The default interval is 3 seconds. To set this feature, use the Connection Retry Delay (ConnectionRetryDelay) connection option.
- Whether the ODBC Client uses client load balancing in its attempts to connect to primary and alternate SequeLink servers. If load balancing is enabled, the ODBC Client uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load balancing. To set this feature, use the Load Balancing (LoadBalancing) connection option. See ["Using Client Load Balancing" on page 81](#) for more information.

On Windows, you can configure a data source to use connection failover on the Failover tab of the driver's Setup dialog box. Refer to the *SequeLink Administrator's Guide* for detailed information.

On UNIX and Linux, you can configure a data source to use connection failover by modifying your system information file (odbc.ini). Refer to the *SequeLink Administrator's Guide* for detailed information.

On Linux, UNIX, and Windows, you can use a connection string to direct the driver to use connection failover. Refer to the *SequeLink Administrator's Guide* for detailed information.

Connection Failover Properties

[Table 1-2](#) summarizes the connection properties that control how connection failover works with the ODBC driver. See [Table 1-1, "ODBC Connection Attributes," on page 61](#) for details about configuring each property.

Table 1-2. Summary: Connection Failover Attributes for the ODBC Driver

Connection Attribute	Characteristic
AlternateServers	<p>A list of alternate SequeLink servers to which the driver will attempt to connect if the primary SequeLink server is unavailable. A port number and an IP address or server name identifying each server are required.</p> <p>If the primary SequeLink server is an LDAP server, each alternate server must be an LDAP server.</p>
ConnectionRetryCount	Number of times the driver retries the primary SequeLink server, and if specified, alternate servers until a successful connection is established. The default is 5.
ConnectionRetryDelay	Wait interval, in seconds, between connection retry attempts when the ConnectionRetryCount property is set to a positive integer. The default is 1.

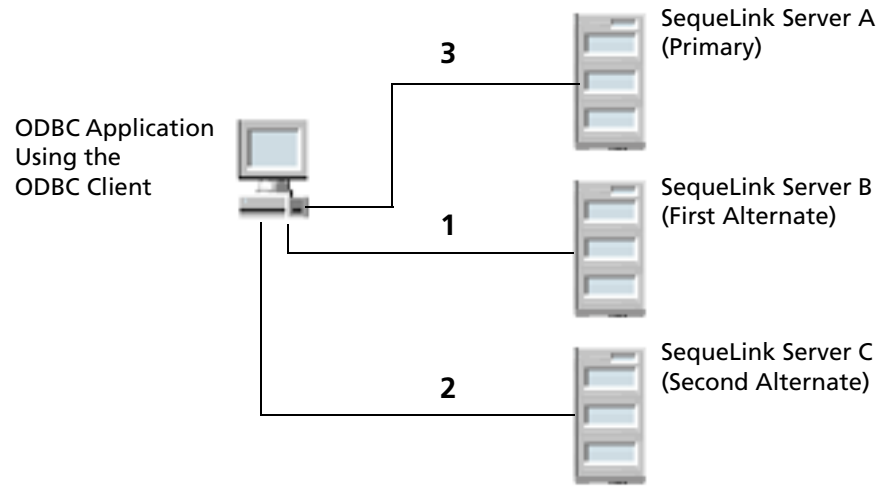
Table 1-2. Summary: Connection Failover Attributes for the ODBC Driver *(cont.)*

Connection Attribute	Characteristic
Host	<p>The TCP/IP address or server name of primary SequeLink server.</p> <p>LDAP: If LDAP is enabled, this identifies the TCP/IP address of the LDAP server. This can also be a list of LDAP servers separated by a blank space (for example, "ld1.foo.com ld2.foo.com ld3.foo.com"). If the first LDAP server in the list does not respond, the ODBC Client will try to connect to the next LDAP server in the list.</p>
LoadBalancing	<p>Sets whether the driver will use client load balancing in its attempts to connect to the list of SequeLink servers (primary and alternate). If client load balancing is enabled, the driver uses a random pattern instead of a sequential pattern in its attempts to connect. The default is false (client load balancing is disabled).</p>
Port	<p>The TCP/IP port on which the primary SequeLink server is listening.</p> <p>If LDAP is enabled, this identifies the TCP/IP port on which the LDAP server is listening.</p>

Using Client Load Balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When client load balancing is enabled, the order in which primary and alternate database servers are tried is random. For example, let us suppose that client load balancing is enabled as shown in [Figure 1-2](#):

Figure 1-2. Client Load Balancing with the ODBC Client



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

For details on configuring client load balancing, refer to the *SequeLink Administrator's Guide*.

Using Connection Retry

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

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

Using connection options, you can specify the number of times the driver attempts to connect and the time in seconds between connection attempts. For details on configuring connection retry, see ["Configuring Connection Failover" on page 76](#).

2 Developing ODBC Applications

This chapter provides information about developing ODBC applications for SequeLink environments, including:

- [“Required ODBC Libraries and Header Files” on page 84](#)
- [“Compiler Requirements” on page 84](#)
- [“ODBC API Functions” on page 85](#)
- [“SQL Escape Sequences” on page 90](#)
- [“Data Types and Isolation Levels” on page 91](#)
- [“Threading” on page 91](#)
- [“Using Scrollable Cursors” on page 93](#)
- [“Using Stored Procedures with Oracle” on page 95](#)
- [“Specifying Application IDs” on page 99](#)
- [“Persisting a Result Set as an XML Data File” on page 101](#)
- [“Error Handling” on page 103](#)
- [“Developing Performance-Optimized ODBC Applications” on page 105](#)

Required ODBC Libraries and Header Files

To develop ODBC applications, you must install the appropriate ODBC libraries and header files for your target platform, as shown in [Table 2-1](#).

Table 2-1. Sources for Required ODBC Development Tools

	Platform	Required Headers and Libraries
	Windows	Microsoft Open Database Connectivity Software Development Kit (SDK).
	UNIX	The required header files and libraries are shipped with the ODBC Client.

NOTE: We recommend that you obtain the Microsoft ODBC 3.x documentation.

Compiler Requirements

The ODBC Client has specific compiler requirements on Linux, UNIX, and Windows. Applications must be compiled with a compiler that is compatible with the guidelines shown in [Table 2-2](#) and [Table 2-3](#).

Table 2-2. Compiler Requirements for Windows

Windows Platform	Compiler Requirements
32-bit	Microsoft Visual C++ .NET 2003 07.01-18059
64-bit	Microsoft Visual C++ .NET 2005

Table 2-3. Compiler Requirements for Linux and UNIX

UNIX Platform	Compiler Requirements
AIX reentrant	<ul style="list-style-type: none"> ■ 32-bit compiler version: VisualAge C++ Professional 6.0 ■ 64-bit compiler version: VisualAge C++ Professional 6.0
Solaris	<ul style="list-style-type: none"> ■ 32-bit compiler version: SUN Studio 9 ■ 64-bit compiler version: SUN Studio 9
Linux	<ul style="list-style-type: none"> ■ 32-bit compiler version: gcc 3.2.3 ■ 64-bit compiler version: gcc 3.3
HP-UX 11 aCC	32-bit compiler version: HP ANSI C++ B3910B A.03.30

ODBC API Functions

The 32-bit and 64-bit ODBC drivers are ODBC Level 1–compliant, supporting all ODBC Core and Level 1 functions. Most Level 2 functions are also supported. [Table 2-4](#) and [Table 2-5](#) list supported functions for ODBC 2.x and ODBC 3.x applications, respectively.

- On Linux and UNIX 64-bit platforms, the compiler flag, ODBC64, must be set.
- On Linux and UNIX, the ODBC header files for the DataDirect Driver Manager are installed in the *install_dir/include* directory.
- On Windows, the ODBC header files are installed with the MDAC installation.

Table 2-4. Function Conformance for 2.x ODBC Application

Core Functions

SQLAllocConnect	SQLExecDirect
SQLAllocEnv	SQLExecute
SQLAllocStmt	SQLFetch
SQLBindCol	SQLFreeConnect
SQLBindParameter	SQLFreeEnv
SQLCancel	SQLFreeStmt
SQLColAttributes	SQLGetCursorName
SQLConnect	SQLNumResultCols
SQLDescribeCol	SQLPrepare
SQLDisconnect	SQLRowCount
SQLDrivers	SQLSetCursorName
SQLError	SQLTransact

Level 1 Functions

SQLColumns	SQLParamData
SQLGetConnectOption	SQLPutData
SQLGetData	SQLSetConnectOption
SQLDriverConnect	SQLSetStmtOption
SQLGetFunctions	SQLSpecialColumns
SQLGetInfo	SQLStatistics
SQLGetStmtOption	SQLTables
SQLGetTypeInfo	

Level 2 Functions

SQLDataSources	SQLNumParams
SQLExtendedFetch	SQLParamOptions
SQLMoreResults	SQLSetScrollOptions
SQLNativeSql	

Table 2-5. Function Conformance for 3.x ODBC Applications

SQLAllocHandle	SQLExecute	SQLNumParams
SQLBindCol	SQLExtendedFetch	SQLNumResultCols
SQLBindParam	SQLFetch	SQLParamData
SQLBindParameter	SQLFetchScroll	SQLPrepare
SQLBulkOperations	SQLForeignKeys	SQLPrimaryKeys
SQLCancel	SQLFreeHandle	SQLProcedureColumns
SQLCloseCursor	SQLFreeStmt	SQLProcedures
SQLColAttribute	SQLGetConnectAttr	SQLPutData
SQLColAttributes	SQLGetCursorName	SQLRowCount
SQLColumnPrivileges	SQLGetData	SQLSetConnectAttr
SQLColumns	SQLGetDescField	SQLSetCursorName
SQLConnect	SQLGetDescRec	SQLSetDescField
SQLCopyDesc	SQLGetDiagField	SQLSetDescRec
SQLDataSources	SQLGetDiagRec	SQLSetEnvAttr
SQLDescribeCol	SQLGetEnvAttr	SQLSetPos
SQLDescribeParam ¹	SQLGetFunctions	SQLSetStmtAttr
SQLDisconnect	SQLGetInfo	SQLSpecialColumns
SQLDriverConnect	SQLGetStmtAttr	SQLStatistics
SQLDrivers	SQLGetTypeInfo	SQLTablePrivileges
SQLEndTran	SQLMoreResults	SQLTables
SQLExecDirect	SQLNativeSql	SQLTransact

1. SQLDescribeParam is not supported on Oracle databases. Instead, use the DataSourceEnableDescribeParam service attribute, described in the *SequeLink Administrator's Guide*.

Binding SQL Statements

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

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

The two data types are identified separately using the `SQLBindParameter` API.

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

To ensure interoperability, the SequeLink ODBC driver uses only the parameter binding information provided by the application. Some DBMSs cannot publish dynamic parameter information back to an ODBC driver. For example, both the SQL Server and Oracle query processors can determine that a parameter is an integer. However, the Oracle query processor cannot publish this information back to the SequeLink ODBC driver.

NOTES:

- The SQL data type is determined at prepare time by the database and does not change for the life of the statement. The SQL data type is not dependent on the data being used by the application. For example, it is not valid to bind the SQL type to SQL_NUMERIC with a precision of 15 and a scale of 5, and then bind it on a later execution to a SQL type of SQL_NUMERIC with a precision of 13 and a scale of 3.
- You can implement SQLDescribeParam only when a database system publishes parameter information after prepare time. The ODBC driver returns this information when the application requests it, but depending on the database, performance penalties can be incurred. You can tune this feature through the SequeLink data source service attribute DataSourceDescribeParam. Refer to the *SequeLink Administrator's Guide* for information about service attributes.

Support for Unicode ODBC W (Wide) Function Calls

The ODBC driver fully supports the SQL-W functions.

The ODBC driver automatically determines whether the database is a Unicode database. [Table 2-6](#) summarizes the way that the ODBC driver maps the database data types to the Unicode data types.

Table 2-6. Support for Unicode ODBC W (Wide) Function Calls

Unicode Data Type	Database Data Type			
	DB2	Oracle	SQL Server	Sybase ¹
SQL_WCHAR	Graphic	CHAR	nchar	CHAR, UNICHAR
SQL_WLONGVARCHAR	Long, Vargraphic	CLOB, LONG	ntext	TEXT
SQL_WVARCHAR	Vargraphic	VARCHAR2	nvarchar, sysname	UNIVARCHAR, VARCHAR

1. The Unicode data types are supported if the UTF-8 character set is installed on the Sybase database.

When used with SequeLink Server for ODBC Socket and SequeLink Server for JDBC Socket, the ODBC driver supports the data types of the backend driver used.

When used with SequeLink Server for Informix and SequeLink Server for JDBC Socket, the ODBC driver does not support Unicode data types.

SQL Escape Sequences

See [Appendix A “SQL Escape Sequences” on page 431](#) for information about the SQL escape sequences supported by the ODBC driver.

Data Types and Isolation Levels

The data types and isolation levels supported by the ODBC driver depend on the data store to which you are connecting. See [Appendix B “Data Types and Isolation Levels” on page 451](#) for database-specific information about data types and isolation levels.

Threading

The ODBC specification requires that all ODBC drivers must be thread-safe; that is, they must not fail when database requests are made on separate threads.

Threading Architecture

An ODBC driver can be based on one of the following architectures:

- *Not thread-safe.* The ODBC driver should not be used in a multi-threaded environment.
- *Thread-impaired.* The ODBC driver serializes all ODBC calls. All requests are handled one by one, without concurrent processing.
- *Thread per connection.* The ODBC driver processes requests concurrently with statement handles that do not share the same connection handle; however requests on the same connection are serialized.
- *Fully threaded.* All requests fully use the threaded model. The ODBC driver processes all requests on multiple statements concurrently.

The ODBC driver supports multithreaded applications on all platforms.

Cancelling Functions in Multithreaded Applications

In a multithreaded application, the application can cancel a function that is running synchronously on a statement. To cancel the function, the application calls `SQLCancel` with the same statement handle as that used by the target function, but on a different thread. Whether `SQLCancel` actually cancels the running function depends on the data store being accessed as shown in [Table 2-7](#).

- *OK* means that `SQLCancel` can interrupt the running function.
- *Ignored* means that `SQLCancel` will have no affect on the running function.

In both cases, `SQLCancel` returns `SQL_SUCCESS`. If `SQLCancel` has been called from a different thread while there is a pending request, the original statement will return `SQL_ERROR` with the error message `Operation cancelled`.

Table 2-7. Using `SQLCancel` in Multithreaded Applications

Data Store	SQLCancel
DB2 UDB for z/OS	Ignored
DB2 UDB on Windows	OK
DB2 UDB on Linux and UNIX	Ignored
Informix	OK
Microsoft SQL Server	OK
Oracle on Windows	Ignored
Oracle on Linux and UNIX	OK
Sybase	OK

Using Scrollable Cursors

Scrollable cursors can move backward and forward in a result set, allowing the application user to scroll back and forth through requested data. SequeLink supports two types of scrollable cursors—static and keyset-driven.

Static and Keyset-Driven Cursors

A *static cursor* is one that does not detect any changes made to the record after the cursor is opened. For example, if a static cursor fetches a row and another application then updates that row, the values would be unchanged when that row is fetched again. A *keyset-driven cursor* detects value changes to the record using keys that are saved when the cursor is opened to retrieve the current data values for each row.

Support for the keyset-driven cursors depends on the data store to which you are connecting, as described in [Table 2-8](#).

Table 2-8. Support for Keyset-Driven Cursors (ODBC)

Database	Conditions for Support
DB2 UDB	The DB2 tables must contain an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.
Informix	None (inherently supported).
Microsoft SQL Server	The table must contain an identity column.
JDBC Socket	The backend database must support an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.

Table 2-8. Support for Keyset-Driven Cursors (ODBC) *(cont.)*

Database	Conditions for Support
ODBC Socket	The backend database must support an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.
Oracle	None (inherently supported).
Sybase	The table must contain an identity column.

Using Static Scrollable Cursors

- The ODBC driver supports static cursors for all types of result set generating statements, including result sets generated by stored procedures.
- The ODBC driver supports LOB data for static cursors; however, by default, only the first 4096 bytes of the LOB column is buffered. See the `SLKStaticCursorLongColBuffLen` connection attribute in [Table 1-1 “ODBC Connection Attributes” on page 61](#) for more information about specifying the amount of data that is buffered.
- To persist a result set as an XML data file with embedded schema, you must use static cursors. See [“Persisting a Result Set as an XML Data File” on page 101](#) for more information about persisting XML data files.

Using Keyset-Driven Scrollable Cursors

- The ODBC driver does not support using keyset-driven cursors on stored procedures or explicit batches.
- The ODBC driver cannot use keyset-driven cursors, when the Select statement contains any of the following SQL language constructions:
 - JOIN
 - Aggregate functions
 - GROUP BY
 - (Informix only). When a fragmented table is not explicitly created with the WITH ROWIDS clause, the ODBC driver returns SQL_SUCCESS_WITH_INFO, with the message that a static cursor was substituted.

Using Stored Procedures with Oracle

SequeLink supports stored procedures against Oracle, including stored procedures in packages.

NOTE: Stored procedures in packages must be qualified with the package name, for example, EmployeePackage.EmployeeProc.

Also, SQLProcedures and SQLProcedureColumns can return information on procedures within PL/SQL packages, allowing ODBC applications to execute these procedures. This section contains an example that shows you how to fetch rows using Oracle PL/SQL procedures.

Example - Part 1

```

Create or replace package EmployeeInfo as
  Type EmployeeRec is record
  (
    Employee_Id      integer,
    Employee_Name    varchar2(25),
    Employee_Job      varchar2(25),
    Department_Name  varchar2(30),
    Employee_Salary  integer
  );
  Type EmployeeCursor is ref cursor return
  EmployeeRec;
End EmployeeInfo;

Create or replace procedure EmployeeInfoProc
(empname IN varchar2, empcursor IN OUT
EmployeeInfo.EmployeeCursor)
As
Begin
  Open empcursor For
  select empno, ename, job, dname, sal
  from emp, dept
  where emp.deptno=dept.deptno and
  ename like empname;
End;
```

NOTE: In this Oracle PL/SQL package, a record type and a cursor (result set) type is defined. The procedure contains an input parameter that can have a value, such as `Smi%`, to request information about employees whose last name starts with the letters 'Smi' (for example, Smith or Smithwick). The procedure also has one input/output parameter of the cursor type defined in the package.

Example - Part 2

This example shows an ODBC function call sequence executing the stored procedure.

```
SQLPrepare(..., '{call EmployeeInfoProc(?)}', ...)
    <- ODBC SQL syntax to executed stored procedures
SQLBindParameter(..., 'Smi%', ...)
    <- Define the input variable for the input marker ?
    in the SQL stmt and assign the value 'Smi%' to it
SQLExecute()
    <- Execute the stored procedure
SQLBindCol()
    <- Assign storage for result column 1 in the
    result set (Employee_Id)
SQLBindCol()
    <- Assign storage for result column 2 in the
    result set (Employee_Name)
SQLBindCol()
    <- Assign storage for result column 3 in the
    result set (Employee_Job)
SQLBindCol()
    <- Assign storage for result column 4 in the
    result set (Department_Name)
SQLBindCol()
    <- Assign storage for result column 5 in the
    result set (Employee_Salary)
SQLFetch()
    <- Fetch the first record from the result set
    generated by the stored procedure.
```

IMPORTANT: From the following procedure definition, you might think that, by having two parameters, the procedure must call `SQLBindParameter` twice:

```
Create or replace procedure EmployeeInfoProc
(empname IN varchar2, empcursor IN OUT
EmployeeInfo.EmployeeCursor)
```

Actually, it does not. The only way to create a result set from an Oracle stored procedure is to declare this result set, `empcursor`, as an input/output parameter. This can be seen in the result of `SQLProcedureColumns(..., 'EmployeeInfoProc', ...)` which an application can use to query the server about a stored procedure.

The following is an excerpt of a session using the tool ODBCtest:

SQLAllocStmt:

In: hdbc=0x004609F0, phstmt=VALID

Return: SQL_SUCCESS=0

SQLPrepare:

In: hstmt=#3 0x00305850, szSqlStr={call EmployeeInfoProc(?)}, cbSqlStr=26

Return: SQL_SUCCESS=0

SQLBindParameter:

In: hstmt=#3 0x00305850, ipar=1, fParamType=SQL_PARAM_INPUT=1,

fCType=SQL_C_CHAR=1,

fSqlType=SQL_CHAR=1, cbColDef=10, ibScale=0, rgbValue=VALID,

cbValueMax=300, pcbValue=VALID, SQL_LEN_DATA_AT_EXEC=FALSE

Return: SQL_SUCCESS=0

SQLExecute:

In: hstmt=#3 0x00305850

Return: SQL_SUCCESS=0

Get Data All:

"EMPNO", "ENAME", "JOB", "DNAME", "SAL"

7934, "MILLER", "CLERK", "ACCOUNTING", 1300.00

7654, "MARTIN", "SALESMAN", "SALES", 1250.00

2 rows fetched from 5 columns.

SQLProcedureColumns:

In: hstmt=#4 0x00305BD8, ...Qualifier=NULL, ...Qualifier=0,

Owner=SCOTT, ...Owner=5, ...Name=EMPLOYEEINFPROC,

...Name=16, ...Name=NULL, ...Name=0

Return: SQL_SUCCESS=0

Get Data All:

"PROCEDURE_CAT", "PROCEDURE_SCHEM", "PROCEDURE_NAME", "COLUMN_NAME",

"COLUMN_TYPE", "DATA_TYPE", ..."TYPE_NAME", "COLUMN_SIZE",

"BUFFER_LENGTH", "DECIMAL_DIGITS", "NUM_PREC_RADIX", "NULLABLE",

"REMARKS", "COLUMN_DEF", "SQL_DATA_TYPE", "SQL_DATETIME_SUB",

"CHAR_OCTET_LENGTH", "ORDINAL_POSITION", "IS_NULLABLE"

", "SCOTT", "EMPLOYEEINFPROC", "EMPNAME", 1, 12, "VARCHAR2", 2000,

2000, <Null>, <Null>, 1, <Null>, <Null>, ...12, <Null>, 2000, 1, "YES"

Specifying Application IDs

Application IDs are alphanumeric strings passed by a SequeLink Client that identify the client application to a SequeLink service that has been configured to accept connections only from specific application IDs.

For more information about configuring SequeLink services to accept connections only from specific application IDs, refer to the *SequeLink Administrator's Guide*.

Specifying Application IDs Explicitly

ODBC client applications can identify themselves explicitly to the SequeLink service in any of the following ways:

- **Specifying the application ID in the ODBC connection string that is passed to `SQLDriverConnect`.** For example:

```
....;APPID=MyAppID;
```

or

```
....;ApplicationID=MyAppID;
```

where *MyAppID* is the application ID.

- **Specifying the application ID using `SQLSetConnectAttr`.** Immediately after each call to `SQLConnect` or `SQLDriverConnect` connecting to the ODBC Client, call `SQLSetConnectAttr` as shown:

```
SQLSetConnectAttr(hdbc, 1053, "myAppId", SQL_NTS)
```

where *myAppId* is the application ID.

The `SQLSetConnectAttr` is defined in `sql.h`. If an incorrect application ID is specified, the `SQLSetConnectAttr` fails and all subsequent SQL statements fail.

Generating Application IDs Automatically

ODBC client applications can turn on automatic application ID generation in any of the following ways:

- **Specifying the automatic application ID method in the ODBC connection string that is passed to `SQLDriverConnect`.** For example:

```
....;AutomaticApplicationID=x;
```

where *x* is set to one of the following values:

- When set to 1, the full path of the application executable is used as input for the hash function.
 - When set to 2, the executable binary file is used as input for the hash function.
 - When set to 3, both the full path of the application executable and the executable binary file are used as input for the hash function.
 - When set to 4, the full directory name of the application executable is used as input for the hash function.
- **Specifying `SQLSetConnectAttr`.** Immediately after each call to `SQLConnect` or `SQLDriverConnect` connecting to the ODBC Client, call `SQLSetConnectAttr` as shown:

```
SQLSetConnectAttr(hdbc, 1054, x, SQL_IS_INTEGER)
```

where *x* is one of the following values:

- When set to 1, the full path of the application executable is used as input for the hash function.
- When set to 2, the executable binary file is used as input for the hash function.

- When set to 3, both the full path of the application executable and the executable binary file are used as input for the hash function.
- When set to 4, the full directory name of the application executable is used as input for the hash function.

Sending Arrays of Parameters

The ODBC driver supports sending arrays of parameters for a parameterized statement. Because the data for a single statement is sent in a single packet, network traffic is reduced.

See [“Using SQLPrepare/SQLExecute and SQLExecDirect” on page 115](#) for more information how sending arrays of parameters can improve performance.

Persisting a Result Set as an XML Data File

The ODBC 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)
```

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

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

2 Execute a SQL statement. For example:

```
SQLExecDirect (hstmt, "Select * from GTABLE", SQL_NTS)
```

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

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

NOTE: A 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 QESQLEXT.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
```

Error Handling

The following types of errors can occur when you are using the ODBC Client:

- ODBC driver errors
- SequeLink Client errors
- SequeLink Server errors
- Database errors

SequeLink® *for* ODBC Driver Errors

An error generated by the ODBC driver has the following format:

```
[DataDirect] [ODBC SequeLink driver] message
```

For example:

```
[DataDirect] [ODBC SequeLink driver] Invalid precision  
specified.
```

If you receive this type of error, check the last ODBC call your application made. Contact your ODBC application vendor, or refer to the ODBC documentation included in the ODBC SDK.

SequeLink® Client Errors

An error generated by the ODBC Client has the following format:

```
[DataDirect] [ODBC SequeLink driver] [SequeLink Client]  
message
```

For example:

```
[DataDirect] [ODBC SequeLink driver] [SequeLink Client] The
specified transliteration module is not found.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

SequeLink® Server Errors

An error generated by SequeLink Server has the following format:

```
[DataDirect] [ODBC SequeLink driver] [SequeLink Server]
message
```

For example:

```
[DataDirect] [ODBC SequeLink driver] [SequeLink Server]
Only Select statements are allowed in this read-only
connection.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

Database Errors

An error generated by the database has the following format:

```
[DataDirect] [ODBC SequeLink driver] [...] message
```

For example:

```
[DataDirect] [ODBC SequeLink driver] [Oracle]
ORA-00942:table or view does not exist.
```


Use the native error code to look up details about the possible cause of the error. For these details, refer to your database documentation.

Developing Performance-Optimized ODBC Applications

This section provides general guidelines for optimizing system performance that have been compiled by examining how numerous shipping ODBC applications have been implemented. These guidelines are divided into the following categories:

- [“Catalog Functions” on page 105](#)
- [“Retrieving Data” on page 110](#)
- [“Selecting ODBC Function” on page 115](#)
- [“Managing Connections and Updates” on page 120](#)

Catalog Functions

Because catalog functions, such as those listed here, are slow compared to other ODBC functions, their frequent use can impair system performance:

- | | |
|-----------------------|----------------------|
| ■ SQLColumns | ■ SQLSpecialColumns |
| ■ SQLColumnPrivileges | ■ SQLStatistics |
| ■ SQLForeignKeys | ■ SQLTables |
| ■ SQLProcedures | ■ SQLTablePrivileges |
| ■ SQLProcedureColumns | |

Minimizing the Use of Catalog Functions

Compared to other ODBC functions, catalog functions are relatively slow. By caching information, applications can avoid multiple executions. Although it is almost impossible to write an ODBC application 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 unions to return the required result set for a single call to a catalog function. These particular elements of the SQL language are performance expensive.

Applications should cache information from catalog functions so that multiple executions are unnecessary. For example, call `SQLGetTypeInfo` once in the application and cache the elements of the result set that your application depends on. It is unlikely that any application uses all elements of the result set generated by a catalog function, so the cached information should not be difficult to maintain.

Avoiding Search Patterns

Passing null arguments or search patterns to catalog functions generates time-consuming queries. In addition, network traffic potentially increases because of unwanted results. Always supply as many non-null arguments to catalog functions as possible. Because catalog functions are slow, applications should invoke them efficiently. Any information that the application can send the driver when calling catalog functions can result in improved performance and reliability.

For example, consider a call to `SQLTables` where the application requests information about the table "Customers." Often, this

call is coded as shown, using the fewest non-null arguments necessary for the function to return success:

```
rc = SQLTables (NULL, NULL, NULL, NULL, "Customers", SQL_NTS, NULL);
```

A driver may process this SQLTables call into SQL as shown:

```
SELECT ... FROM SysTables WHERE TableName = 'Customers' UNION ALL
SELECT ... FROM SysViews WHERE ViewName = 'Customers' UNION ALL
SELECT ... FROM SysSynonyms WHERE SynName = 'Customers'
ORDER BY ...
```

In this example, the application provided little information about the object for which information was requested. 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 may 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 improve. Less network traffic would be required to return only one result row and unwanted rows would be filtered by the database.

In addition, if the TableType argument can be supplied, 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'
```

Determining Table Characteristics with a Dummy Query

Avoid using `SQLColumns` to determine table characteristics. Instead, use a dummy query with `SQLDescribeCol`.

Consider an application that allows the user to choose the columns that will be selected. Should the application use `SQLColumns` to return information about the columns to the user or prepare a dummy query and call `SQLDescribeCol`?

Case 1: `SQLColumns` Method

```
rc = SQLColumns (... "UnknownTable" ...);
// This call to SQLColumns will generate a query to
// the system catalogs... possibly a join which must be
// prepared, executed, and produce a result set
rc = SQLBindCol (...);
rc = SQLExtendedFetch (...);
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained
```

Case 2: `SQLDescribeCol` Method

```
// prepare dummy query
rc = SQLPrepare (... "SELECT * from UnknownTable
    WHERE 1 = 0" ...);
// query is never executed on the server - only prepared
rc = SQLNumResultCols (...);
for (irow = 1; irow <= NumColumns; irow++) {
    rc = SQLDescribeCol (...)
    // + optional calls to SQLColAttributes
}
// result column information has now been obtained
// Note we also know the column ordering within the table!
// This information cannot be
// assumed from the SQLColumns example.
```

In both cases, a query is sent to the server. In Case 1, the query must be evaluated and form a result set that is returned to the client. Case 2 is the better performing model.

To complicate this discussion, let us consider a database server that does not natively support preparing a SQL statement. The performance of Case 1 does not change, but the performance of Case 2 improves slightly, because the dummy query is evaluated before being prepared. Because the Where clause of the query always evaluates to FALSE, the query generates no result rows and is processed without accessing table data. Again, Case 2 outperforms Case 1.

Managing the Retrieval of Database Meta-Information

Meta-information is information that describes the data stored in the database and can include information about the tables in the database, the columns in those tables, and the indexes that are defined for those tables. This data also is referred to as the database's *data dictionary* or *system catalog*.

Typically, ODBC applications extract and use information from the database's data dictionary using specific calls, such as the ODBC calls `SQLTables`, `SQLColumns`, and `SQLPrimaryKeys`. In large databases, the amount of meta-information that is retrieved can be considerable. Because some client applications cannot manage large amounts of information efficiently, system performance can be adversely affected.

Some ODBC calls have parameters that accept search patterns. You can use these parameters to limit the amount of meta-information that is retrieved; however, not every client application supports these parameters.

SequeLink allows you to use database data dictionary filters and database data dictionary views to limit the amount of meta-information that is retrieved.

Using Database Data Dictionary Filters

Database Data Dictionary filters limit the amount of meta-information that can be retrieved from the database's native data dictionary. Specifically, they limit the number of result rows that can be returned for SQLTables. The data dictionary filters override any call parameters that are passed by the application when it accesses the database's native data dictionary.

SequeLink provides the following types of database data dictionary filters, which must be defined on the server:

- Filter by catalog list
- Filter by schema list
- Filter by table type
- Filter by database (DB2 for z/OS only)

For more information about setting the database data dictionary filters for a SequeLink service, refer to DataSourceSchemaFilterList in the *SequeLink Administrator's Guide*.

z/OS Using Database Data Dictionary Views

The DataSourceDB2CatalogOwner service attribute on the server allows you to limit the meta-information that is returned by using views on the database data dictionary. Database Data Dictionary views are supported for DB2 for z/OS only. For more information about setting the DataSourceDB2CatalogOwner service attribute, refer to the *SequeLink Administrator's Guide*.

Retrieving Data

To retrieve data efficiently, return only the data that you need, and choose the most efficient method of doing so. The guidelines in this section will help you optimize system performance when retrieving data with ODBC applications.

Retrieving Long Data

Unless it is necessary, applications should not request long data (SQL_LONGVARCHAR and SQL_LONGVARBINARY data), because retrieving long data across a network is slow and resource-intensive.

Most users do not want to see long data. If the user does need to see these result items, the application can query the database again, specifying only the long columns in the select list. This method allows the average user to retrieve result sets 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 (for example, some applications `SELECT * FROM table_name ...`). If the select list contains long data, the driver must retrieve that data at fetch time, even if the application does not bind the long data in the result set. When possible, use a method that does not retrieve all columns of the table.

Reducing the Size of Retrieved Data

To reduce network traffic and improve performance, you can reduce the size of data being retrieved to some manageable limit by calling `SQLSetStmtOption` with the `SQL_ATTR_MAX_LENGTH` option. This reduces network traffic and improves performance.

Although 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 , 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, (SQLLEN)
    sizeof(CaseContainer), ...);
```

At this point, it is more likely that an ODBC driver will retrieve 64 KB of information from the server, instead of 1000 bytes. In terms of network access, one 64-KB retrieval is less expensive than 64 retrievals of 1000 bytes. Unfortunately, the application might not call `SQLGetData` again; therefore, the first and only retrieval of `CaseHistory` would be slowed by the fact that 64 KB of data must be sent across the network.

Many ODBC drivers allow you to limit the amount of data retrieved across the network by supporting the `SQL_MAX_LENGTH` attribute. This attribute allows the driver to communicate to the database server that only x bytes of data are relevant to the client. The server responds by sending only the first x bytes of data for all result columns. This optimization substantially reduces network traffic and improves performance. The previous example returned only one row, but, consider the case where 100 rows are returned in the result set—the performance improvement would be substantial.

Using Bound Columns

Retrieving data using bound columns (SQLBindCol), instead of SQLGetData, reduces the ODBC call load and improves performance.

Consider the following example:

```
rc = SQLExecDirect (hstmt, "SELECT <20 columns>
    FROM Employees WHERE HireDate >= ?", SQL_NTS);
do {
rc = SQLFetch (hstmt);
// call SQLGetData 20 times
} while ((rc == SQL_SUCCESS) || (rc==
SQL_SUCCESS_WITH_INFO));
```

Suppose the query returns 90 result rows. In this case, 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 how SQLBindCol is used by binding result information directly from the database server to the user's buffer. That is, instead of the driver retrieving information into a container and 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` instead of `SQLFetch` to retrieve data. 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 previous example using `SQLExtendedFetch`, instead of `SQLFetch`:

```
rc = SQLSetStmtOption (hstmt, SQL_ROWSET_SIZE, 100);
// use arrays of 100 elements
rc = SQLExecDirect (hstmt, "SELECT <20 columns>
    FROM Employees WHERE HireDate >= ?", SQL_NTS);
// call SQLBindCol 1 time specifying row-wise binding
do {
rc = SQLExtendedFetch (hstmt, SQL_FETCH_NEXT, 0,
    &RowsFetched, RowStatus);
} while ((rc == SQL_SUCCESS) || (rc ==
SQL_SUCCESS_WITH_INFO));
```

Notice the improvement from the previous examples. The initial call load was more than 1890 ODBC calls. By choosing ODBC calls carefully, the number of ODBC calls made by the application has now been reduced to 4 (1 `SQLSetStmtOption` + 1 `SQLExecDirect` + 1 `SQLBindCol` + 1 `SQLExtendedFetch`). In addition to reducing the call load, many ODBC drivers retrieve data from the server in arrays, further improving 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`). Although using the cursor library does not improve performance, it should not be detrimental to application response time when using forward-only cursors (no logging is required). Furthermore, using the cursor library when `SQLExtendedFetch` is not supported natively by the ODBC driver simplifies the code because the

application can depend on `SQLExtendedFetch` being available. The application does not require two algorithms (one using `SQLExtendedFetch` and another using `SQLFetch`).

Choosing the Right Data Type

Advances in processor technology brought significant improvements to the way that operations such as floating-point math are handled; however, retrieving and sending certain data types are still expensive when the active portion of your application will not fit into on-chip cache. When you are working with data on a large scale, it is still important to select the data type that can be processed most efficiently. For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

Processing time is shortest for character strings, followed by integers, which usually require some conversion or byte ordering. Processing floating-point data and timestamps is at least twice as slow as processing integers.

Selecting ODBC Function

The guidelines in this section help you to selecting which ODBC functions will give you the best performance.

Using SQLPrepare/SQLExecute and SQLExecDirect

Using `SQLPrepare/SQLExecute` is not always as efficient as using `SQLExecDirect`. Use `SQLExecDirect` for queries that will be

executed once and SQLPrepare/SQLExecute for queries that will be executed multiple times.

ODBC drivers are optimized based on the perceived use of the functions that are being executed. SQLPrepare/SQLExecute is optimized for multiple executions of statements that use parameter markers. SQLExecDirect is optimized for a single execution of a SQL statement. Unfortunately, more than 75% of all ODBC applications use SQLPrepare/SQLExecute exclusively.

Consider an ODBC driver that implements SQLPrepare by creating a stored procedure on the server which contains the prepared statement. Creating stored procedures involves substantial overhead, but the statement can be executed multiple times. Although creating stored procedures is performance-expensive, processing is minimal because the query is parsed and optimization paths are stored at create procedure time.

Using SQLPrepare/SQLExecute for a statement that is executed only once results in unnecessary overhead. Furthermore, applications that use SQLPrepare/SQLExecute for large single execution query batches exhibit poor performance. Similarly, applications that always use SQLExecDirect do not perform as well as those that use a logical combination of SQLPrepare/SQLExecute and SQLExecDirect sequences.

Using Arrays of Parameters

Passing arrays of parameter values for bulk insert operations, for example, with SQLPrepare/SQLExecute and SQLExecDirect can reduce the ODBC call load and network traffic. To use arrays of parameters, the application calls SQLSetStmtAttr with the following attribute arguments:

- `SQL_ATTR_PARAMSET_SIZE` sets the array size of the parameter.

- `SQL_ATTR_PARAMS_PROCESSED_PTR` assigns a variable filled by `SQLExecute`, which contains the number of rows that are actually inserted.
- `SQL_ATTR_PARAM_STATUS_PTR` points to an array in which status information for each row of parameter values is returned.

NOTE: With ODBC 3.x, calls to `SQLSetStmtAttr` with the `SQL_ATTR_PARAMSET_SIZE`, `SQL_ATTR_PARAMS_PROCESSED_ARRAY`, and `SQL_ATTR_PARAM_STATUS_PTR` arguments replace the ODBC 2.x call to `SQLParamOptions`.

Before executing the statement, the application sets the value of each data element in the bound array. When the statement is executed, the driver tries to process the entire array contents using one network round trip. For example, let us compare the following examples, Case 1 and Case 2.

Case 1: Executing Prepared Statement Multiple Times

```
rc = SQLPrepare (hstmt, "INSERT INTO DailyLedger (...)
VALUES (?, ?, ...)", SQL_NTS);
// bind parameters
...
do {
// read ledger values into bound parameter buffers
...
rc = SQLExecute (hstmt);
// insert row
} while ! (eof);
```

Case 2: Using Arrays of Parameters

```
SQLPrepare (hstmt, " INSERT INTO DailyLedger (...) VALUES
(?, ?, ...)", SQL_NTS);
SQLSetStmtAttr (hstmt, SQL_ATTR_PARAMSET_SIZE, (UDWORD)100,
SQL_IS_INTEGER);
SQLSetStmtAttr (hstmt, SQL_ATTR_PARAMS_PROCESSED_PTR,
&rows_processed, SQL_IS_POINTER);
```

```

// Specify an array in which to return the status of
// each set of parameters.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAM_STATUS_PTR,
ParamStatusArray, SQL_IS_POINTER);
// pass 100 parameters per execute
// bind parameters
...
do {
// read up to 100 ledger values into
// bound parameter buffers
...
rc = SQLExecute (hstmt);
// insert a group of 100 rows
} while ! (eof);

```

In Case 1, if there are 100 rows to insert, 101 network round-trips are required to the server, one to prepare the statement with `SQLPrepare` and 100 additional round-trips for each time `SQLExecute` is called.

In Case 2, the call load has been reduced from 100 `SQLExecute` calls to only 1 `SQLExecute` call. Furthermore, network traffic is reduced considerably.

Using SQLPrepare and Multiple SQLExecute Calls

Applications that use `SQLPrepare` and multiple `SQLExecute` calls should use `SQLParamOptions`. Passing arrays of parameter values reduces the ODBC call load and network traffic.

Consider the following example that inserts 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, `SQLExecute` is called 100 times, resulting in 100 network requests to the server.

Alternatively, consider 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 is reduced from 100 to just 2 `SQLExecute` calls. Furthermore, network traffic is reduced considerably. To achieve the best performance, applications should contain algorithms for using `SQLParamOptions`. `SQLParamOptions` is ideal for copying data into new tables or bulk loading tables. Note, however, that some ODBC drivers do not support `SQLParamOptions`.

Using the Cursor Library

If the driver provides scrollable cursors, do not use the cursor library automatically. The cursor library creates local temporary log files, which are performance-expensive to generate and provide worse performance than native scrollable cursors.

The cursor library adds support for static cursors, which simplifies the coding of applications that use scrollable cursors. However, the cursor library creates temporary log files on the user's local disk drive as it performs the task. Typically, disk input/output is a

slow operation. Although the cursor library is beneficial, applications should not choose automatically to use the cursor library when an ODBC driver supports scrollable cursors natively.

Typically, ODBC drivers that support scrollable cursors achieve better performance by requesting that the database server produce a scrollable result set, instead of emulating the capability by creating log files. Many applications use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_ODBC);
```

but should use:

```
rc = SQLSetConnectOption (hdbc, SQL_ODBC_CURSORS,  
    SQL_CUR_USE_IF_NEEDED);
```

Managing Connections and Updates

The guidelines in this section will help you to manage connections and updates to improve system performance for your ODBC applications.

Managing Connections

Connection management affects application performance. Optimize your applications by connecting once and using multiple statement handles, instead of performing multiple connections. Avoid connecting to a data source after establishing an initial connection.

Although gathering driver information at connection is a good practice, it often is more efficient to gather it in one step rather than two steps. Some ODBC applications are designed to call informational gathering routines that have no record of already attached connection handles. For example, some applications establish a connection and then call a routine in a separate DLL or shared library that reattaches and gathers information about the

driver. Applications that are designed as separate entities should pass the already connected HDBC pointer to the data collection routine instead of establishing a second connection.

Another bad practice is to connect and disconnect several times throughout your application to perform SQL statements. Connection handles can have multiple statement handles associated with them. Statement handles can provide memory storage for information about SQL statements. Therefore, applications do not need to allocate new connection handles to process SQL statements. Instead, applications should use statement handles to manage multiple SQL statements.



On Windows, you can significantly improve performance with connection pooling, especially for applications that connect over a network or through the World Wide Web. With connection pooling, closing connections does not close the physical connection to the database. When an application requests a connection, an active connection from the connection pool is reused, avoiding the network input/output needed to create a new connection.

Connection and statement handling should be addressed before implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

Managing Commits in Transactions

Committing data is extremely disk input/output intensive and slow. If the ODBC driver can support transactions, always turn Autocommit off.

What does a commit actually involve? The database server must flush back to disk every data page containing updated or new data. This is not a sequential write, but a searched write to replace existing data in the table. By default, Autocommit is on when connecting to a data source, and Autocommit mode

usually impairs performance because of the significant amount of disk input/output required to commit every operation.

Some database servers do not provide an Autocommit mode. For this type of server, the ODBC driver must explicitly issue a COMMIT statement and a BEGIN TRANSACTION for every operation sent to the server. In addition to the large amount of disk input/output required to support Autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

Choosing the Right Transaction Model

Many systems support distributed transactions; that is, transactions that span multiple connections. Distributed transactions are at least four times slower than normal transactions due to the logging and network input/output necessary to communicate between all the components involved in the distributed transaction. Unless distributed transactions are required, avoid using them. Instead, use local transactions when possible.

Using Positional Updates and Deletes

Use positional updates and deletes or SQLSetPos to update data. Although positional updates do not apply to all types of applications, developers should use positional updates and deletes when it makes sense. Positional updates (using “update where current of cursor” or using SQLSetPos) allow the developer to signal the ODBC driver to “change the data here” by positioning the database cursor to the appropriate row to be changed. The designer is not forced to build a complex SQL statement and simply supplies the data that will be changed.

In addition to making the application more easily maintainable, positional updates usually result in improved performance. Because the database server is already positioned on the row for

the Select statement in process, performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server usually has an internal pointer to the row available (for example, ROWID).

Using SQLSpecialColumns

Use SQLSpecialColumns to determine the optimal set of columns to use in the Where clause for updating data. Often, pseudo-columns provide the fastest access to the data, and these columns can only be determined by using SQLSpecialColumns.

Some applications cannot be designed to take advantage of 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 may formulate the Where clause by using all searchable result columns or by calling SQLStatistics to find columns that may be part of a unique index. These methods usually work, but may result in fairly complex queries.

Consider the following:

```
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name,
    ssn, address, city, state, zip FROM emp", SQL_NTS);
// fetchdata
...
rc = SQLExecDirect (hstmt, "UPDATE EMP SET ADDRESS = ?
    WHERE first_name = ? and last_name = ? and ssn = ?
    and address = ? and city = ? and state = ? and zip = ?",
    SQL_NTS);
// fairly complex query
```

Applications should call SQLSpecialColumns/SQL_BEST_ROWID to retrieve the optimal set of columns (possibly a pseudo-column) that identifies a specific record. Many databases support special columns that are not explicitly defined by the user in the table definition but are “hidden” columns of every table (for example, ROWID and TID). These pseudo-columns provide the fastest

access to data because they typically point to the exact location of the record. Because pseudo-columns are not part of the explicit table definition, they are not returned from `SQLColumns`. To determine if pseudo-columns exist, call `SQLSpecialColumns`.

Consider the previous example again:

```
...
rc = SQLSpecialColumns (hstmt, ..... 'emp', ...);
...
rc = SQLExecDirect (hstmt, "SELECT first_name, last_name,
    ssn, address, city, state, zip, ROWID FROM emp",
    SQL_NTS);
// fetch data and probably "hide" ROWID from the user
...
rc = SQLExecDirect (hstmt, "UPDATE emp SET address = ?
    WHERE ROWID = ?", SQL_NTS);
// fastest access to the data!
```

If your data source does not contain special pseudo-columns, the result set of `SQLSpecialColumns` consists of the columns of the optimal unique index on the specified table (if a unique index exists); therefore, your application does not need to call `SQLStatistics` to find the smallest unique index.

Part 2: Developing ADO Applications

This part contains the following chapters:

- [Chapter 3 “Using the ADO Client” on page 127](#) provides information about using ADO applications with the SequeLink Client *for* ADO.
- [Chapter 4 “Developing ADO Applications” on page 171](#) provides information about developing ADO applications for SequeLink environments.

3 Using the ADO Client

This chapter provides information about using ADO/OLE DB applications with the *SequeLink Client for ADO* (the ADO Client).

About the ADO Client

The ADO Client supports ADO/OLE DB applications through a component called the *SequeLink for ADO data provider*. The ADO Client is an ADO middleware data access component. It uses ADO technology to connect business applications to relational data stores (like Oracle). In most cases, minimal user interaction with the ADO Client middleware is needed because it works in the background, providing connectivity transparently.

Some businesses write and support their own applications for data access. For example, application developers may need to write ADO or OLE DB-based data consumers that use the ADO provider. This book provides programming information for developers who want to control data source connections from within their applications.

After you install the ADO Client, your OLE DB- or ADO-based business applications (data consumers) can automatically detect it on your system. Depending on the data consumer's design, the data consumer can connect directly to a data source that uses the ADO provider, or it can provide a way to select the data provider to make a connection.

You use the DataDirect Configuration Manager to define ADO data sources for the ADO provider. After you have defined data sources that use the provider, you can select the provider from your data consumer to make a connection. See [“Configuring ADO Client Data Sources” on page 133](#) for instructions on creating and configuring data sources.

After you have defined ADO data sources, you can access them from your data consumer and connect to them. See [“Testing ADO Connections” on page 147](#) for more information.

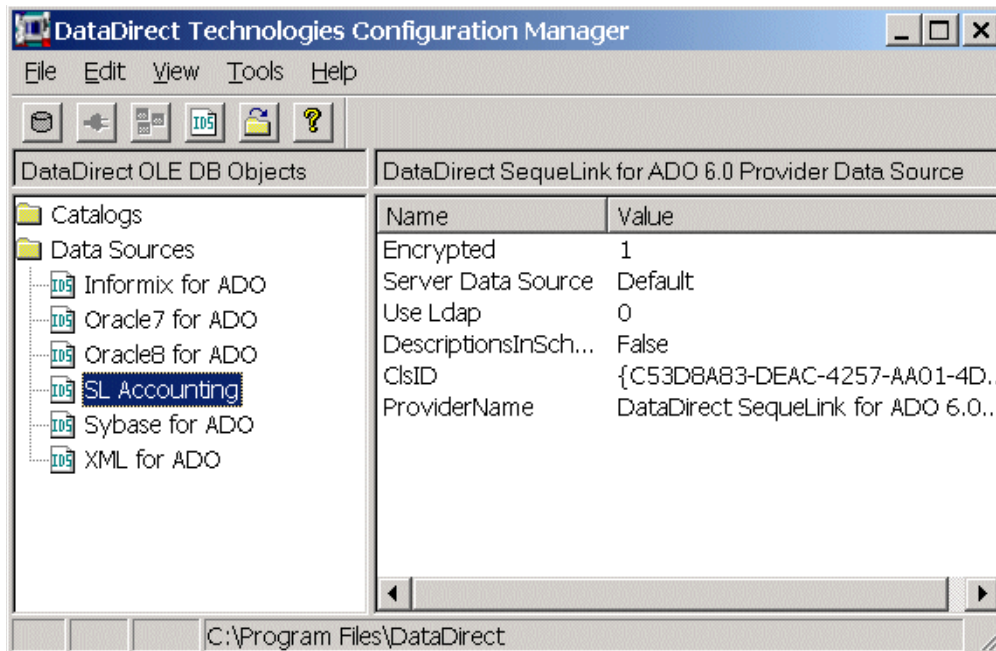
Using the DataDirect Configuration Manager

To create and configure data sources for the ADO Client, you use the DataDirect Technologies Configuration Manager.

To start the Configuration Manager, select **Start / Programs**, and select **DataDirect SequeLink 6.0 Client for ADO**. Then, select the **DataDirect Configuration Manager** application.

The Configuration Manager window is divided into two panes. As [Figure 3-1](#) shows, the left pane displays a folder containing defined ADO data sources. When you select a data source, the right pane displays the properties for the selected data source.

Figure 3-1. DataDirect Technologies Configuration Manager






Double-click the **Data Sources** folder to display any existing ADO data sources. The Configuration Manager displays the ADO data sources contained in the current directory, which is shown in the status bar at the bottom of the Configuration Manager. The first time you start the Configuration Manager, the current directory defaults to the \Program Files\DataDirect\slado60 directory.

Working with the DataDirect Configuration Manager

Table 3-1 summarizes the parts and functions of the Configuration Manager that you use with ADO data sources.

NOTE: Options that are not supported by the ADO provider are disabled in the toolbar and are omitted from this description.

Table 3-1. DataDirect Technologies Configuration Manager: Parts and Functions for ADO Data Sources		
Use this element...	To do this...	
Toolbar		Create new data sources
		Change the current directory
		View online help
Menu Bar	File	■ Create a new data source
		■ Exit from the DataDirect Configuration Manager
	Edit	■ Delete a data source
		■ Rename a data source
	View	■ Modify a data source
		■ View or hide the toolbar and status bar
		■ Refresh the Configuration Manager

Shortcut Tip: Right-clicking an item in the left pane displays a pop-up menu that allows you to perform the same actions that are available from the toolbar and menu bar.

Table 3-1. DataDirect Technologies Configuration Manager: Parts and Functions for ADO Data Sources (cont.)

Use this element...	To do this...
Tools	<ul style="list-style-type: none"> ■ Change the directory in which to look for data sources ■ Define a Template data source directory ■ Define a Master data source directory
Help	View online help.
Vertical splitter bar	Adjust the size of the left and right panes.
Status bar	<ul style="list-style-type: none"> ■ Show the current keyboard state, including when NUM LOCK, SCROLL LOCK, and CAPS LOCK are turned on ■ Show the current directory

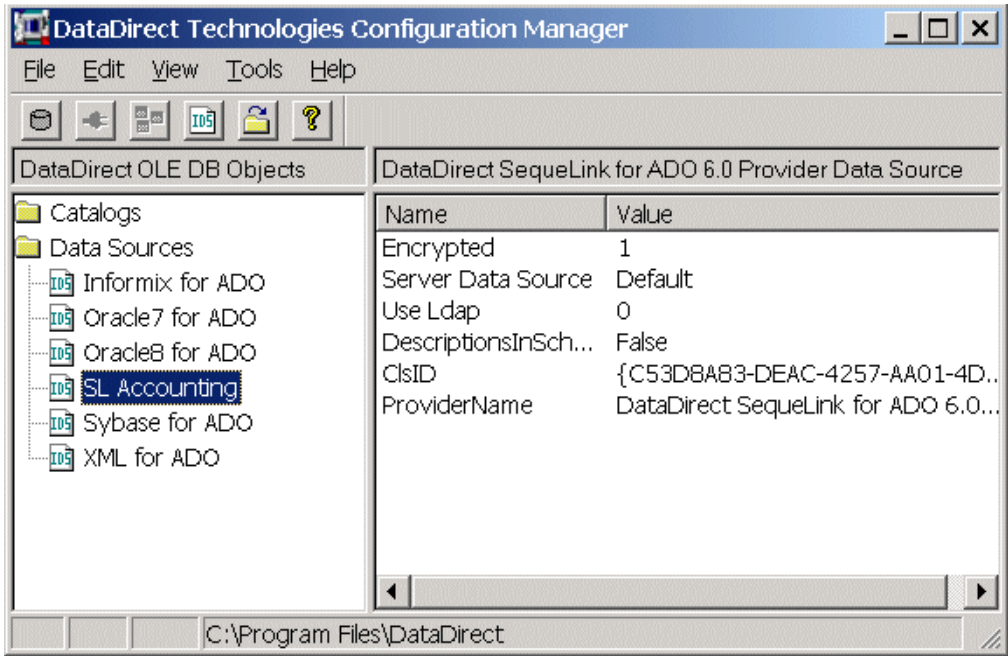
Shortcut Tip: Right-clicking an item in the left pane displays a pop-up menu that allows you to perform the same actions that are available from the toolbar and menu bar.

Displaying Data Source Properties

- 1 Start the Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **DataDirect SequeLink 6.0 Client for ADO**. Then, select the **DataDirect Configuration Manager** application.
- 2 Double-click the **Data Sources** folder to display any existing ADO data sources.
- 3 Highlight a data source in the list. The properties of the data source display in the right pane. For example, the following

figure shows the properties of an ADO data source named SL Accounting displayed in the right pane.

Figure 3-2. DataDirect Technologies Configuration Manager: Displaying Data Source Properties



You can right-click a data source in the left pane to display a pop-up menu. The pop-up menu offers the same actions for the item that are available from the Edit menu.

To display a setup window for an existing data source, double-click an ADO data source in the Data Sources folder.

To create a new data source, highlight the **Data Sources** folder; then, select **File / New / Data Source** from the menu bar.

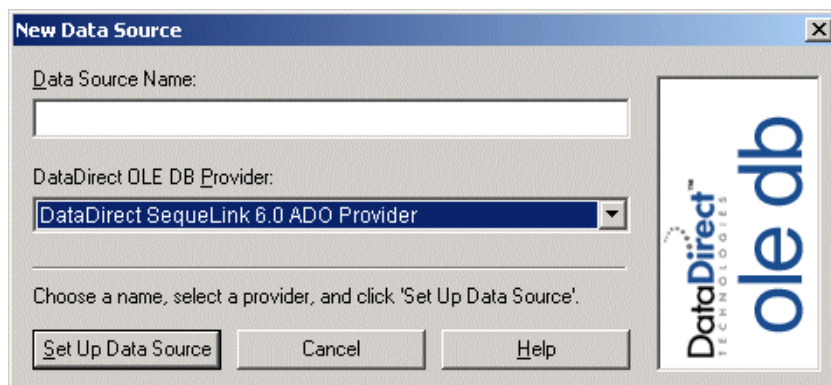
Configuring ADO Client Data Sources

The following sections provide instructions for configuring ADO client data sources:

- ["Creating an ADO Client Data Source" on page 133](#)
- ["Modifying an ADO Client Data Source" on page 140](#)
- ["Renaming an ADO Client Data Source" on page 141](#)
- ["Deleting an ADO Client Data Source" on page 141](#)
- ["Copying an ADO Client Data Source" on page 142](#)
- ["Changing Data Source Directories" on page 143](#)
- ["Defining Default Setup Options" on page 143](#)

Creating an ADO Client Data Source

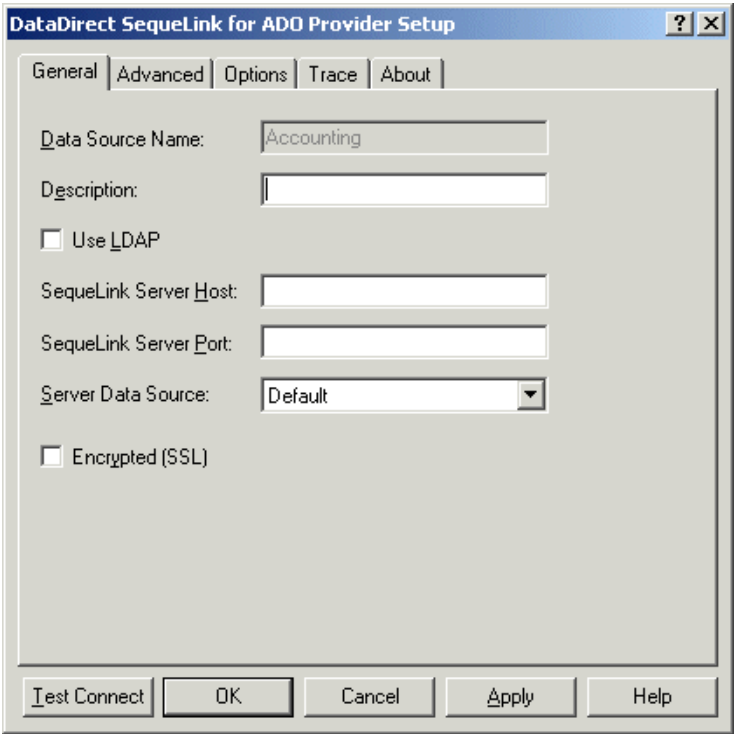
- 1 Start the DataDirect Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **DataDirect SequeLink 6.0 for ADO Client**. Then, select the **DataDirect Configuration Manager** application.
- 2 Select **File / New / Data Source** from the menu bar. The New Data Source window appears.



- 3 Type a name for the data source. All data sources located in the same directory must have unique names. If the name has

already been used for another data source, you are prompted to enter a different name.

- 4 In the DataDirect OLE DB Providers drop-down list, select **DataDirect SequeLink 6.0 ADO Provider**. Then, click **Set Up Data Source**.
- 5 The DataDirect SequeLink ADO Provider Setup window appears.



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

- 6 Provide the following information.
Data Source Name: This is a read-only field that uniquely identifies this ADO data source configuration. Examples include Accounting or SequeLink to Oracle Data.

Description: Optionally, type a description of the data source. For example, *My Accounting Database* or *Accounting Data in Oracle*.

SequeLink Server Host: Type the TCP/IP host name of the SequeLink service to which you want the ADO Client to connect. This field is available only if the Use LDAP check box is **not** selected.

SequeLink Server Port: Type the TCP/IP port the SequeLink service is listening on for incoming connection requests. The port you specify must be the same as the one that was specified for the SequeLink service when the SequeLink Server was installed; the default is 19996. This field is available only if the Use LDAP check box is **not** selected.

Server Data Source: Type the name of a server data source configured for the SequeLink service to use for the connection or select one from the drop-down list. This field is optional. If a server data source is not specified, the default server data source for that SequeLink service will be used for the connection. This field is available only if the Use LDAP check box is **not** selected.

Use LDAP: To configure the ADO Client to retrieve connection information from an LDAP directory, select the **Use LDAP** check box. The fields change on the lower half of the screen to accommodate the information that is required to query an LDAP server for connection information. Provide the following information:

LDAP Server Host: Type the TCP/IP host name of the LDAP server.

LDAP Server Port: Type the TCP/IP port on which the LDAP server is listening for incoming connection requests. The default value is 389.

Distinguished Name (DN): Type an identifier that uniquely identifies the LDAP entry where connection information is stored.

For more information about retrieving connection information from LDAP directories, refer to the *SequeLink Administrator's Guide*.

Encrypted (SSL): If the remote SequeLink service is configured for SSL encryption, select this check box. You must select this check box if connecting to a SequeLink service enabled for SSL.

When the check box is cleared (the default), communication between the SequeLink Client and SequeLink Server is not encrypted with SSL.

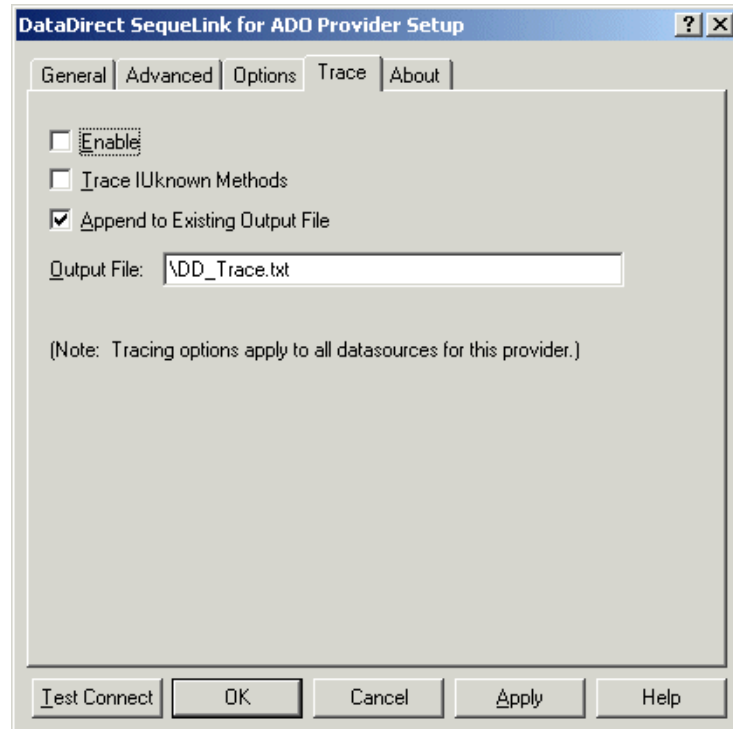
For more information about encrypting data, refer to the *SequeLink Administrator's Guide*.

NOTES:

- Data encryption with SSL is not supported for LDAP Servers. The Use LDAP and the Encrypted (SSL) check boxes are mutually exclusive.
- SSL encryption is not supported for SequeLink Server for DB2 on z/OS. To support SSL in a DB2 for z/OS environment, use the SequeLink Proxy Server. For information about using the SequeLink Proxy Server, refer to the *SequeLink Administrator's Guide*.

- 7 Optionally, click the **Trace** tab to enable tracing options. Specify values on the Trace tab, then, click **Apply**.

NOTE: Settings on this tab apply to all data sources for the ADO data provider. You cannot set the trace options programmatically.



Enable: Select this check box to enable tracing support. By default, the check box is not selected.

Trace IUnknown Methods: Select this check box to enable tracing support of IUnknown methods. By default, the check box is not selected.

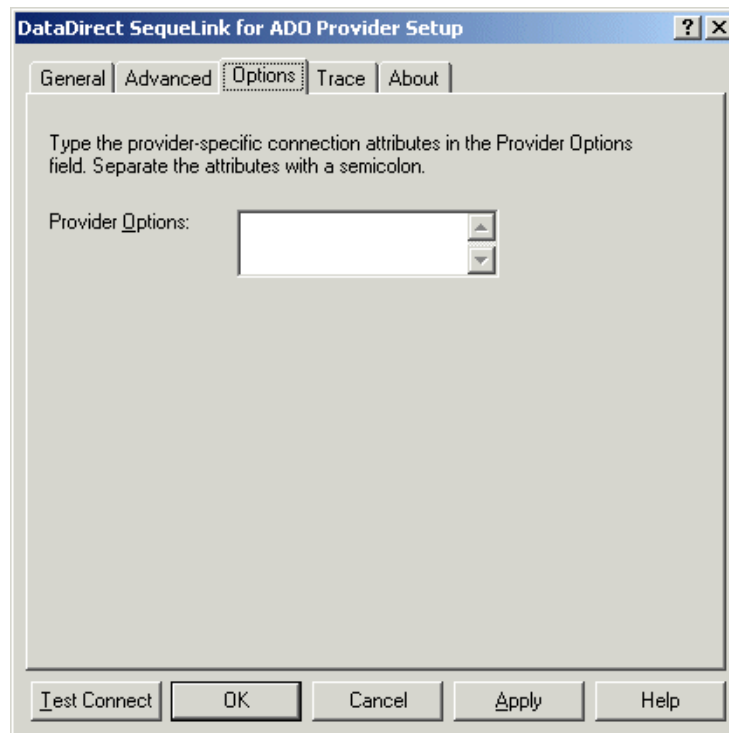
Append to Existing Output File: Select this check box to append tracing results to an output file. By default, the check box is selected.

Output File: Type the name of the file to which tracing results will be appended. This file contains the tracing results of all data sources for the ADO data provider.

- 8 Optionally, click the **Options** tab to add connection attributes (see [“Connecting with a Provider String” on page 155](#) for the values that can be entered on this tab). Values in the Provider Options field are separated by semicolons.

For example, the following string sets values for the Alternate Servers, Connection Retry Count, and Connection Retry Delay connection failover options:

```
Alternate Servers=(Host=server2:Port=19996,Host=server3:Port=19996,Host=server4:Port=19996);Connection  
Retry Count=2;Connection Retry Delay=3
```



- 9 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 provider Setup dialog box. A logon dialog box appears; see [“ADO Connection Dialogs” on page 148](#) for details.

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

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

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

NOTE: All data sources are saved to the current directory displayed in the Configuration Manager. See [“Changing Data Source Directories” on page 143](#) for instructions on changing the current directory.

Modifying an ADO Client Data Source

To modify the properties of a data source, double-click the data source in the Data Sources folder of the Configuration Manager to display the SequeLink for ADO Provider Setup window. See [“Creating an ADO Client Data Source” on page 133](#) for a description of the fields you can change.

Renaming an ADO Client Data Source

You can rename data sources. You cannot rename or delete the Data Sources folder.

To rename an ADO provider data source:

- 1 Start the Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **DataDirect SequeLink 6.0 Client for ADO**. Then, select the **DataDirect Configuration Manager** application.
- 2 Select the data source you want to rename.
- 3 Select **Edit / Rename**. The data source name becomes an editable field.
- 4 Type the new name of the data source and press ENTER.

Deleting an ADO Client Data Source

- 1 Start the Configuration Manager. To start the Configuration Manager, select **Start / Programs**, and select **DataDirect SequeLink 6.0 Client for ADO**. Then, select the **DataDirect Configuration Manager** application.
- 2 Select the data source you want to delete.
- 3 Select **Edit / Delete**.
- 4 A window appears prompting you to confirm the deletion. Click **Yes** to delete the selected data source.

After you change the current directory, the left pane of the Configuration Manager is automatically refreshed to display the data sources in the new directory.

The current directory remains active until you change it again. Any data sources you create are saved to the current directory.

Copying an ADO Client Data Source

Copying a data source can make it easier for you to configure new data sources that use the same properties as existing data sources. When you copy a data source, the copied data source retains all the properties of the original data source. After copying, you can modify the properties of the data source as needed.

To copy a data source:

- 1 In Windows Explorer, navigate to the directory that contains the data source you want to copy. All ADO provider data sources use .IDS as their file extension. For example, if the data source name appears as TEST in the Configuration Manager, the name of the data source file is TEST.IDS.

NOTE: The directory location of a data source displayed in the Configuration Manager appears in the status bar at the bottom of the Configuration Manager.
- 2 Copy the data source to the Windows Explorer clipboard; then, perform one of the following actions:
 - To copy to a different directory, navigate to the directory you want to copy to and paste the data source in that new directory. You can use the same data source name.
 - To copy to the same directory, paste the data source; then, rename the data source to a *unique* name.
- 3 To display the new data source in the Configuration Manager, perform one of the following actions:
 - If you copied the data source to a different directory, make that directory the current directory in the Configuration Manager by selecting **Tools / Options / Main Data Source Directory**. The new data source appears in the Data Sources folder.

- If you copied the data source to the same directory and renamed the data source, select **View / Refresh** in the Configuration Manager. The new data source appears in the Data Sources folder.

Changing Data Source Directories

The Configuration Manager displays the ADO data sources contained in the current directory, which is displayed in the status bar at the bottom of the Configuration Manager. The first time you start the Configuration Manager, the current directory defaults to the ADO Client installation directory.

To change the current directory:



- 1 Click the **Change main Data Source directory** button on the tool bar.
- 2 Type the name of the new directory in the Current Directory field, or, click the **Browse** button to select a different directory.
- 3 Click **OK**.

After you change the current directory, the left pane of the Configuration Manager is automatically refreshed to display the data sources in the new directory. The current directory remains active until you change it again. Any data sources you create are saved to the current directory.

Defining Default Setup Options

The Configuration Manager supports configurable default setup options and override options through the use of a template data source file and a master data source file, respectively.

A template data source file is used by the Configuration Manager to populate values in the fields of the Setup dialog box when a user creates a new data source. By creating a template data source file, you can define the default setup options (default values for newly created data sources). The user can change these default values when setting up a new data source.

A master data source file is used to provide global connection options. The options set in the master data source file override connection options set any other way (for example, by the data source specified by an application or a connection string) when an application is connecting to the database.

Creating a Template Data Source File

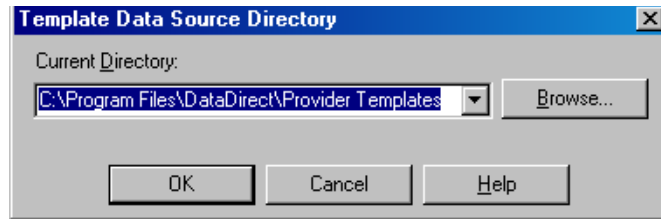
You can define template data source files to simplify the creation of data source files. A template data source file allows you to define the default setup options for SequeLink data providers. The Configuration Manager supplies these values in the Setup dialog box when a user creates a new data source. The user can change these default values when setting up a new data source.

To create a template data source file:

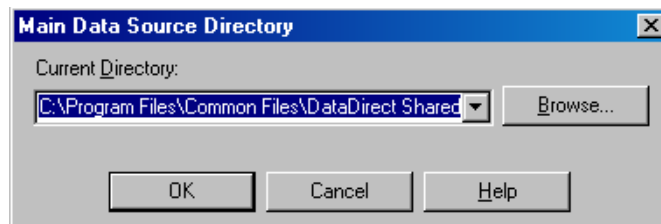
- 1 Create a directory in which to store the template data source file.

IMPORTANT: The template data source directory cannot be the same as the directory for other data sources.

- 2 In the Configuration Manager window, select **Tools / Options / Template Data Source Directory**. Specify the directory that you created in Step 1; then, click **OK**.



- 3 Select **Tools / Options / Main Data Source Directory**. Specify the template directory; then, click **OK**. This sets the template directory as the location in which to create the template data source file.



- 4 Create a data source, defining the values that will be most commonly used. This will be your template data source file for the specified data provider.
- 5 Select **Tools / Options / Main Data Source Directory**. Specify the directory that contains your data sources; then, click **OK**.

Creating a Master Data Source File

You can define a master data source file that overrides connection options set any other way. This allows you to control the way that users connect to the database.

During connection, the Main data source directory is checked for a data source, and connection values are retrieved. If a Master data source directory exists, it is then checked for the same data

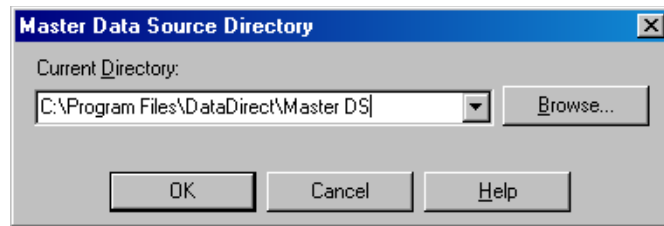
source. The connection settings for user data sources will be overridden by the master data source file.

To create a master data source file:

- 1 Create a directory in which to store the master data source file.

IMPORTANT: The master data source directory cannot be the same as the directory for template data sources or any other data provider data sources.

- 2 In the Configuration Manager window, select **Tools / Options / Master Data Source Directory** and specify the directory that you created in Step 1. The master data source file will be used at connection time.



- 3 Select **Tools / Options / Main Data Source Directory**. Specify the master data source directory; then, click **OK**. This sets the master data source directory as the location in which to create the master data source file.
- 4 Create one or more data sources. The data sources in this directory will be your master data source files for the specified data providers.
- 5 Select **Tools / Options / Main Data Source Directory**. Specify the directory that contains your data sources; then, click **OK**.

Connecting to an ADO Client

You can connect to a data source using a Connection window, or using a provider string. See [“Connecting with a Provider String” on page 155](#) for information about connecting using an ADO provider string.

Testing ADO Connections

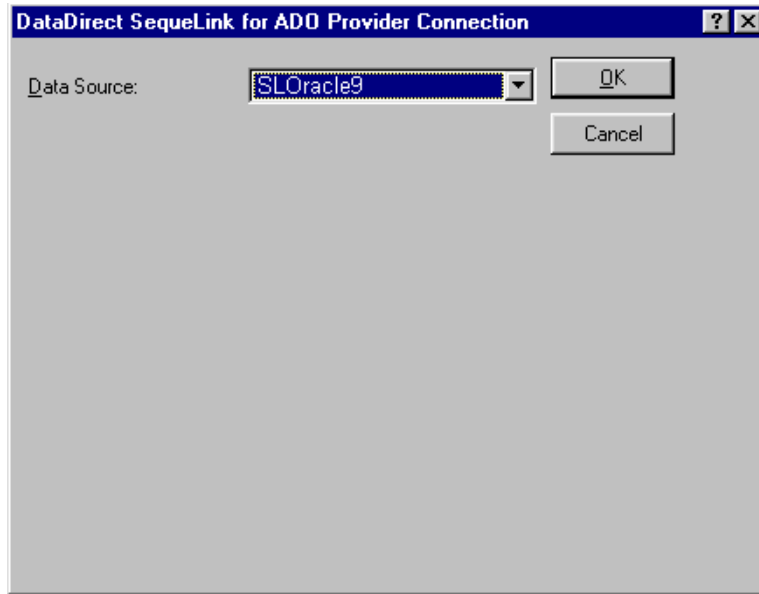
The ADO provider opens a Connection window when you perform either of the following actions:

- You request a connection to an ADO provider from within your data consumer, and your data consumer requests the ADO provider to prompt for missing connection parameters.
- You click **Test Connect** in an ADO provider setup window to test the connection to a data source you have set up.

See [“ADO Connection Dialogs” on page 148](#) for more information about ADO connection dialogs that may appear.

ADO Connection Dialogs

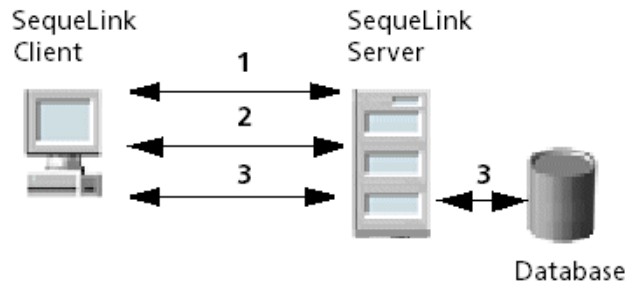
When your data consumer requests the ADO provider to prompt for missing connection parameters and an ADO data source has not been specified, the DataDirect SequeLink for ADO Provider Connection dialog box appears.



Select the data source that you want to use from the drop-down list. If you do not want to specify a data source name, select **None** from the drop-down list. In some cases, the data source name may be supplied automatically. Then, click **OK**.

The other connection dialogs that may appear involve prompting for information required to make a SequeLink data access connection.

A SequeLink data access connection involves the following stages:



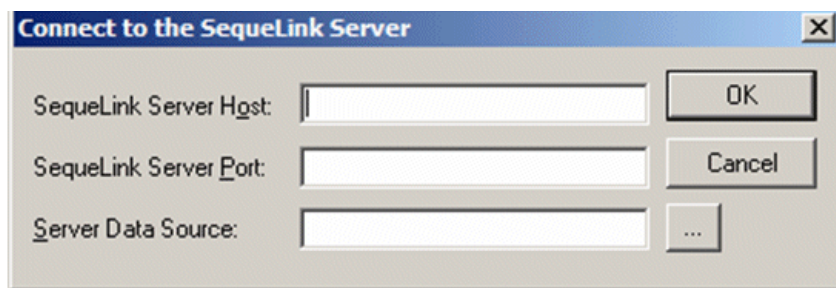
- 1 A network connection is established.
- 2 An authentication mechanism is used to establish the identity of the SequeLink Client to the SequeLink Server.
- 3 Based on information provided by the SequeLink Client application (for example, a database user name and password), a database connection is established.

Stage 1: Establishing a Network Connection

The first stage of the connection process involves establishing a network connection. The dialog that appears depends on whether the connection has been configured to connect directly to a SequeLink service or to retrieve connection information for the SequeLink service from a centralized LDAP directory.

Connecting Directly to a SequeLink® Service

If the connection has been configured to connect directly to a SequeLink service, the Connect to the SequeLink Server dialog box appears.



Provide the following information; then, click **OK**.

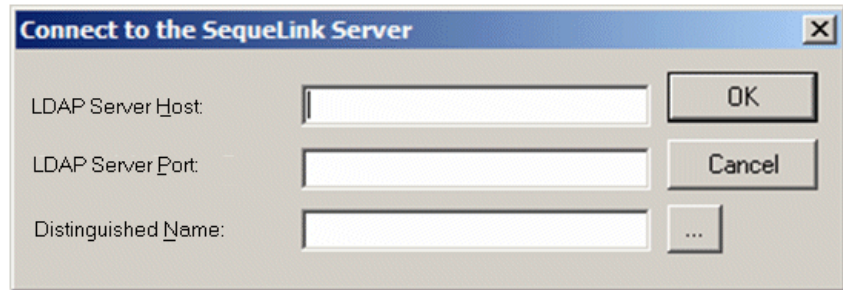
SequeLink Server Host: Type the TCP/IP host name of the SequeLink service.

SequeLink Server Port: Type the TCP/IP port on which the SequeLink service is listening. A default installation of SequeLink Server uses the port 19996.

Server Data Source: Type the name of a server data source to use for the connection or click the ... button to select an existing data source. This step is optional. If a server data source is not specified, the default server data source for that service will be used for the connection.

Retrieving Connection Information from an LDAP Directory

If the connection has been configured to connect to an LDAP server to retrieve connection information from an LDAP directory, the Connect to the SequeLink Server dialog box appears.

A screenshot of a Windows-style dialog box titled "Connect to the SequeLink Server". The dialog box has a blue title bar with a close button (X) in the top right corner. It contains three text input fields: "LDAP Server Host:", "LDAP Server Port:", and "Distinguished Name:". To the right of each field is a corresponding button: "OK" for the first, "Cancel" for the second, and an ellipsis button ("...") for the third. The dialog box has a standard Windows XP-style border and a light gray background.

Provide the following information; then, click **OK**.

LDAP Server Host: Type the TCP/IP host name of the LDAP server.

LDAP Server Port: Type the TCP/IP port on which the LDAP server is listening.

Distinguished Name: Type the Distinguished Name (DN) of the LDAP entry.

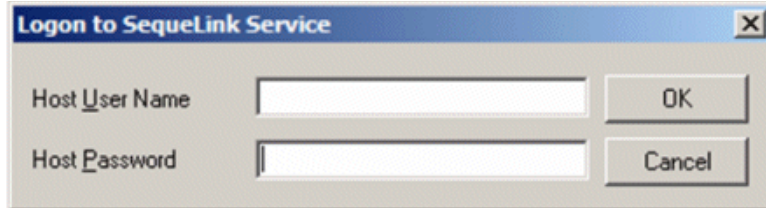
For information about setting up an LDAP server for SequeLink, refer to the *SequeLink Administrator's Guide*.

Stage 2: SequeLink® Server Authentication

The second stage of the connection process involves authentication of the SequeLink Client to the SequeLink Server. The dialog boxes that appear depend on how authentication is configured for the SequeLink service.

- When ServiceAuthMethods=anonymous or ServiceAuthMethods=integrated_nt, no dialogs appear.

- When `ServiceAuthMethods=OSLogon(HUID,HPWD)` or `ServiceAuthMethods=OSLogon(UID,PWD)`, the Logon to SequeLink Service dialog box appears.



Provide the following information; then, click **OK**.

Host User Name: Type the host user name.

NOTE: When connecting to a Windows server, you must prefix the host user name with a server name, if authenticating to a local server, or a domain name (for example, SALES\DJONES). If the server name or domain name is omitted, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the machine on which the SequeLink Server is running. If this validation fails, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the domain of the machine on which the SequeLink Server is running.

Host Password: Type the host password.

- When `ServiceAuthMethods=OSLogon(HUID,HPWD,NPWD)` or `ServiceAuthMethods=OSLogon(UID,PWD,NPWD)` and the password is expired, the Password expired. Please specify new password dialog box appears.

A screenshot of a Windows-style dialog box titled "Password expired. Please specify new password." with a close button (X) in the top right corner. The dialog contains four text input fields: "Host User Name" with the text "mfigp", "Host Password" with masked characters "XXXXXXXX", "New Password", and "Confirm Password". To the right of the "Host Password" field are two buttons: "OK" and "Cancel".

NOTE: If the password is not expired, the previous dialog appears. You are only prompted for the Host User Name and Host Password.

Provide the following information; then, click **OK**.

Host User Name: Type the host user name.

NOTE: When connecting to a Windows server, you must prefix the host user name with a server name, if authenticating to a local server, or a domain name (for example, SALES\DJONES). If the server name or domain name is omitted, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the machine on which the SequeLink Server is running. If this validation fails, the SequeLink Server will attempt to authenticate the user ID and password with the database account defined for the domain of the machine on which the SequeLink Server is running.

Host Password: Type the host password.

New Password: Type the new password to be used by the SequeLink password change mechanism.

Confirm Password: Type again the new password to confirm it.

For more information about configuring authentication, refer to the *SequeLink Administrator's Guide*.

Stage 3: Data Store Logon

The last stage of the connection process involves logging on the data store. The dialogs that appear depend on the data store logon method configured for the SequeLink service:

- When `DataSourceLogonMethod=OSIntegrated`, no dialogs appear.
- When `DataSourceLogonMethod=DBMSLogon(UID,PWD)` or `DataSourceLogonMethod=DBMSLogon(DBUID,DBPWD)`, a data store-specific user name and password are required and the Logon to SequeLink Service dialog box appears.

A screenshot of a Windows-style dialog box titled "Logon to SequeLink Service". The dialog has a blue title bar with a close button (X) in the top right corner. It contains three text input fields: "Database User Name:", "Database Password:", and "Database:". To the right of the "Database User Name:" and "Database Password:" fields are "OK" and "Cancel" buttons, respectively. The "Database:" field is positioned below the "Database Password:" field.

Provide the following information; then, click **OK**.

Database User Name: Type the database logon ID.

Database Password: Type the database password.

Database: Type the name of the database to which you want to connect. This field is disabled when the data store does not recognize the concept of databases.

For more information about configuring data store logon methods, refer to the *SequeLink Administrator's Guide*.

Connecting with a Provider String

Once a data source is defined through the DataDirect Configuration Manager and the SequeLink *for* ADO Provider Setup Assistant, your application can connect directly to that data source. You can override the current settings for the data source when you connect using a *provider string*.

A provider string contains *attribute=value* pairs that control various aspects of the data provider's connection and interaction with the database. When an application names a specific data source to connect to, the application can also pass the data provider a provider string of *attribute=value* pairs. The data provider uses the values in the provider string instead of the default values defined for the data source in the system information.

Using provider strings allows application developers to configure connections for users programmatically and ensures that users have the optimum settings for working with the provider and database. Any values a user has set for a data source through the DataDirect Configuration Manager are overridden by corresponding values in the provider string for the current session only.

The provider string sets the DBPROP_INIT_PROVIDERSTRING initialization property and has the form:

```
"attribute=value;attribute=value;"
```

You can specify the *attribute=value* pairs on the Options tab of the DataDirect SequeLink *for* ADO 6.0 Provider Setup window.

See ["ADO Connection Attributes"](#) for information about the ADO connection attributes.

ADO Connection Attributes

Table 3-2 provides a list of ADO connection attributes supported by the ADO provider. It lists a description for each attribute. The defaults listed in the table are initial defaults that apply when no value is specified in the provider string or in the data source definition in the system information. If you specified a value for the attribute when configuring the data source in the Setup window, that value is your default.

Table 3-2. ADO Connection Attributes

Attribute	Description
Alternate Servers	<p>Specifies a list of alternate SequeLink servers that the data provider will try to connect to if the primary SequeLink server is unavailable. Specifying a value for this attribute enables connection failover for the data provider.</p> <p>The value must be in the form of a string that defines connection information for each alternate SequeLink server. The Host and Port values are required for each alternate server entry. Connection options (<i>property=value</i>) are optional for each alternate server entry. The string has the format:</p> <pre>(Host=servername1:Port=port1[:property=value[:...]],Host=servername2:Port=port2[:property=value[:...]],...)</pre> <p>For example, the following Alternate Servers value defines two alternate SequeLink servers for connection failover:</p> <pre>Alternate Servers=(Host=AccountingSLServer:Port=13999:APPName=SequeLink for ADO App,Host=AccountingAltServer:Port=13998:APPName=SequeLink for ADO App)</pre> <p>IMPORTANT: If you specify an LDAP server in the Host connection attribute, the alternate servers must be LDAP servers. For example, the following Alternate Servers value defines three alternate LDAP servers for connection failover:</p> <pre>Alternate Servers=(Host=ld1.foo.com:Port=389,Host=ld2.foo.com:Port=389,Host=ld3.foo.com:Port=389)</pre> <p>See “Configuring Connection Failover” on page 164 for a discussion of connection failover.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Application ID	<p>Specifies the application ID that identifies the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to is configured to limit access to specific applications.</p> <p>See “Specifying Application IDs” on page 208 for more information about using application IDs to limit access to SequeLink services.</p>
ApplicationName	<p>Identifies the application that is establishing the connections (for example, ApplicationName=Account01) and can be used to identify where problems that are associated with a particular application occur.</p> <p>The initial default value is SequeLink for ADO Application.</p>
Automatic Application ID	<p>Specifies an application ID that is automatically generated by the ADO Client to identify the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications.</p> <p>See “Specifying Application IDs” on page 208 for more information about using application IDs to limit access to SequeLink services.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Connection Retry Count	<p>Specifies the number of times the data provider retries connection attempts to the primary SequeLink server, and if specified, alternate SequeLink servers after the first unsuccessful attempt. Valid values are 0 and any positive integer.</p> <p>When set to 0 (the initial default), the data provider does not try to reconnect after the initial unsuccessful attempt.</p> <p>For example, consider the following example:</p> <pre>Alternate Servers=(Host=server2:Port=19996,Host=server3:Port=19996,Host=server4:Port=19999);Connection Retry Count=1;</pre> <p>If a connection is not successfully established on the data provider's first pass through the list of database servers, the data provider retries all the servers in the list one time.</p> <p>See "Configuring Connection Failover" on page 164 for a discussion of connection failover.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Connection Retry Delay	<p>Specifies the seconds the data provider waits after the initial unsuccessful connection attempt before retrying a connection to the primary SequeLink server and, if specified, to the alternate SequeLink servers.</p> <p>Valid values are integers from 0 to 65535.</p> <p>The default value is 3 (seconds). When set to 0, there is no delay between retries.</p> <p>NOTE: This option has no effect unless the Connection Retry Count connection option is set to an integer value greater than 0.</p> <p>For example, in the following example:</p> <pre>Alternate Servers=(Host=server2:Port=19996,Host=server3:Port=19996,Host=server4:Port=19996);Connection Retry Count=2;Connection Retry Delay=3</pre> <p>If a connection is not successfully established on the data provider's first pass through the list of SequeLink servers, the data provider retries the list of servers twice. It waits 3 seconds between the first and second connection retry attempts.</p> <p>See "Configuring Connection Failover" on page 164 for a discussion of connection failover.</p>
Database	<p>Specifies the name of the database to which you want to connect.</p>
Database User Name	<p>Specifies the data store user name, which may be required depending on the server configuration.</p>
Database Password	<p>Specifies the data store password, which may be required depending on the server configuration.</p>
Data Source	<p>Specifies a string that identifies an ADO/OLE DB data source configuration. Examples include <code>Accounting</code> or <code>SequeLink to Oracle Data</code>.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Default Length for Long Data	<p>Turns on a workaround that allows you to specify the amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with a static cursor.</p> <p>The initial default value is 4.</p>
Distinguished Name	<p>Specifies the distinguished name identifying the LDAP entry from which connection information is retrieved. This attribute is required when UseLDAP=1.</p>
Encrypted	<p>Encrypted={0 1}. Enables the use of SSL encryption if the remote SequeLink service the client is connecting to is configured for SSL.</p> <p>When set to 0 (the default), the data provider does not use SSL encryption for data exchanged with the SequeLink Server.</p> <p>When set to 1, the data provider uses SSL encryption. This attribute must be set to 1 when connecting to a SequeLink service enabled for SSL.</p> <p>For more information about encrypting data, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>NOTE: Encrypted (SSL) is not supported for LDAP Servers. The Use LDAP and the Encrypted (SSL) attributes are mutually exclusive.</p>
Host	<p>Specifies the TCP/IP address of the SequeLink Server, specified in dotted format or as a host name.</p> <p>LDAP: If LDAP is enabled, this identifies the TCP/IP address of the LDAP server. This can also be a list of LDAP servers separated by a blank space (for example, "ld1.foo.com ld2.foo.com ld3.foo.com"). If the first LDAP server in the list does not respond, the data provider will try to connect to the next LDAP server in the list.</p> <p>NOTE: If you want to use connection failover features such as connection retry and load balancing, specify the alternate LDAP servers in the Alternate Servers connection attribute. See "Configuring Connection Failover" on page 164 for more information about connection failover.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Host Password	Specifies the host password, which may be required depending on the server configuration.
Host User Name	Specifies the host user name, which may be required depending on the server configuration.
Load Balancing	<p>Load Balancing={0 1}. Determines whether the data provider uses client load balancing in its attempts to connect to a list of SequeLink servers (primary and alternate). The list of alternate servers is specified by the Alternate Servers attribute.</p> <p>When set to 1, client load balancing is used and the data provider attempts to connect to the list of SequeLink servers (primary and alternate servers) in random order.</p> <p>If set to 0 (the default), client load balancing is not used and the driver connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified).</p> <p>In the following example:</p> <pre>Host=server1;Port=19996;User ID=test;Password=secret;Alternate Servers=(Host=server2:Port=19996,Host=server3:Port=19996, Host=server4:Port=19996);Load Balancing=1;</pre> <p>The data provider randomly selects a SequeLink server from the list of primary and alternate servers and attempts to connect. If that connection attempt fails, the data provider again randomly selects a SequeLink server from this list until all of the servers have been tried or a connection is successfully established.</p> <p>See “Configuring Connection Failover” on page 164 for more information about connection failover.</p>

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
New Password	<p>Specifies the new host password to be used. If specified and applicable to the connection, the SequeLink password change mechanism is invoked. When the password has been changed successfully, the following warning is generated:</p> <pre>[DataDirect][SequeLink ADO Provider] [SequeLink Server] The user password was changed successfully</pre> <p>If unspecified and the SequeLink Server detects that the host password has expired, you will be prompted for a new host password.</p> <p>For more information about the SequeLink password change mechanism, refer to the <i>SequeLink Administrator's Guide</i>.</p>
Password	Specifies the host or data store password, which may be required depending on the server configuration.
Port	<p>Specifies the TCP/IP port on which the SequeLink Server is listening.</p> <p>LDAP: If LDAP is enabled, this identifies the TCP/IP port on which the LDAP server is listening. If you do not specify a port, the default port for LDAP (389) will be used.</p>
Server Data Source	Optionally, identifies the server data source to be used for the connection. If not specified, the configuration of the default server data source will be used for the connection.

Table 3-2. ADO Connection Attributes (cont.)

Attribute	Description
Use LDAP	<p>Use LDAP={0 1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP.</p> <p>When set to 0, the SequeLink Client will connect directly to the specified SequeLink Server.</p> <p>When set to 1, the SequeLink Client will retrieve the TCP/IP host, TCP/IP port, and SequeLink data source (optional) from an LDAP entry identified by a Distinguished Name (DN). Once the connection information is retrieved, the SequeLink Client will connect directly to the specified SequeLink Server. The DistinguishedName (DN) attribute is required.</p> <p>The initial default value is 0.</p> <p>NOTE: Encryption (SSL) is not supported for LDAP Servers. The Use LDAP and the Encrypted (SSL) attributes are mutually exclusive.</p>
User ID	<p>Specifies the host or data store user name, which may be required depending on the server configuration.</p>

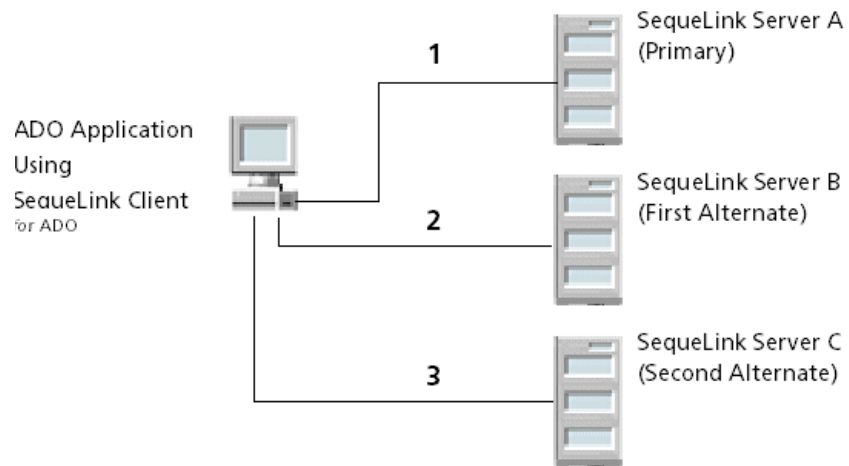
Configuring Connection Failover

Connection failover allows an application to connect to an alternate, or backup, database server if the primary SequeLink server is unavailable, for example, because of a hardware failure or traffic overload. Connection failover ensures that the data on which your critical ADO and OLE DB applications depend is always available.

You can customize the ADO Client for connection failover by configuring a list of alternate SequeLink servers that are tried if the primary server is not accepting connections. Connection attempts continue until a connection is successfully established or

until all the alternate SequeLink servers have been tried the specified number of times.

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



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

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the driver can retry each alternate SequeLink Server (primary and alternate) for a specified number of attempts.

To configure connection failover, you **must** specify a list of alternate SequeLink Servers that are tried at connection time if

the primary server is not accepting connections. To do this, use the Alternate Servers connection property. Connection attempts continue until a connection is successfully established or until all the database servers in the list have been tried once (the default).

Optionally, you can specify the following additional connection failover features:

- The number of times the ADO Client attempts to connect to the primary and alternate SequeLink servers after the initial unsuccessful connection attempt. By default, the driver does not retry. To set this feature, use the Connection Retry Count connection option. See [“Using Connection Retry” on page 169](#) for more information.
- The wait interval, in seconds, between attempts to connect to the primary and alternate SequeLink servers. The default interval is 3 seconds. To set this feature, use the Connection Retry Delay connection option.
- Whether the ADO data Client uses client load balancing in its attempts to connect to primary and alternate SequeLink servers. If load balancing is enabled, the driver uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load balancing. To set this feature, use the Load Balancing connection option. See [“Using Client Load Balancing” on page 168](#) for more information.

Connection Failover Properties

[Table 3-3](#) summarizes the connection properties that control how connection failover works with the ADO Client. See [Table 3-1](#) for details about configuring each property.

Table 3-3. Summary: Connection Failover Properties for the ADO Data Provider

Property	Characteristic
Alternate Servers	<p>A list of alternate SequeLink servers to which the data provider will attempt to connect if the primary SequeLink server is unavailable. A port number and an IP address or server name identifying each server are required.</p> <p>If the primary SequeLink server is an LDAP server, each alternate server must be an LDAP server.</p>
Connection Retry Count	Number of times the data provider retries the primary database server, and if specified, alternate servers until a successful connection is established. The default is 5.
Connection Retry Delay	Wait interval, in seconds, between connection retry attempts when the ConnectionRetryCount property is set to a positive integer. The default is 1.
Host	<p>The TCP/IP address or server name of primary SequeLink server.</p> <p>LDAP: If LDAP is enabled, this identifies the TCP/IP address of the LDAP server. This can also be a list of LDAP servers separated by a blank space (for example, "ld1.foo.com ld2.foo.com ld3.foo.com"). If the first LDAP server in the list does not respond, the ADO Client will try to connect to the next LDAP server in the list.</p>
Load Balancing	Sets whether the ADO Client uses client load balancing in its attempts to connect to the list of database servers (primary and alternate). If client load balancing is enabled, the ADO Client uses a random pattern instead of a sequential pattern in its attempts to connect. The default is 0 (client load balancing is disabled).

Table 3-3. Summary: Connection Failover Properties for the ADO Data Provider
(cont.)

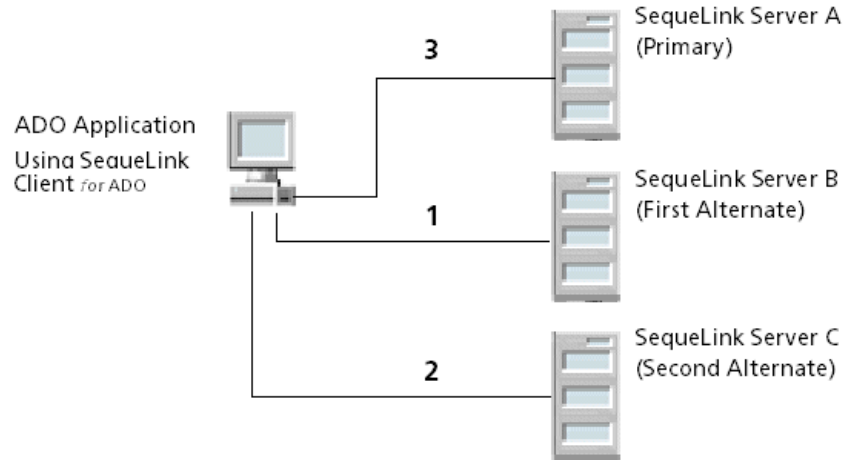
Property	Characteristic
Port	Port listening for connections on the primary SequeLink server. If LDAP is enabled, this identifies the TCP/IP port on which the LDAP server is listening. If you do not specify a port, the default port for LDAP (389) will be used.

See [“Configuring Connection Failover” on page 164](#) and [“Using Client Load Balancing” on page 168](#) for overviews of connection failover and client load balancing.

Using Client Load Balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When client load balancing is enabled, the order in which primary and alternate database servers are tried is

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



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

For details on configuring client load balancing, refer to the *SequeLink Administrator's Guide*.

Using Connection Retry

Connection retry defines the number of times the SequeLink Client attempts to connect to the primary SequeLink Server and, if configured, alternate SequeLink Servers after the initial unsuccessful connection attempt. Connection retry can be an important strategy for system recovery. For example, suppose

you have a power failure in which both the SequeLink Client and the SequeLink Server fail. When the power is restored and all computers are restarted, the SequeLink Client may be ready to attempt a connection before the SequeLink Server has completed its startup routines. If connection retry is enabled, the client application can continue to retry the connection until a connection is successfully accepted by the SequeLink Server.

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

Using connection options, you can specify the number of times the driver attempts to connect and the time in seconds between connection attempts. For details on configuring connection retry, see [“Configuring Connection Failover” on page 164](#).

4 Developing ADO Applications

This chapter provides information about developing ADO applications for SequeLink environments including:

- [“OLE DB Objects and Interfaces” on page 172](#)
- [“Supported Schema Rowsets” on page 174](#)
- [“Supported OLE DB Property Groups” on page 175](#)
- [“OLE DB Interfaces Supported in ADO” on page 188](#)
- [“Mapping ADO Methods and Properties” on page 190](#)
- [“Data Shaping” on page 206](#)
- [“Persisting Information” on page 207](#)
- [“Using Rowsets” on page 207](#)
- [“Mapping Data Types” on page 208](#)
- [“Specifying Application IDs” on page 208](#)
- [“Error Handling” on page 209](#)

OLE DB Objects and Interfaces

The ADO data provider supports Insert, Update, and Delete operations through the OLE DB Rowset interfaces and through the command interfaces (using SQL DML).

Table 4-1 lists the OLE DB objects that the ADO data provider supports, and the interfaces implemented for each object.

Table 4-1. Objects and Interfaces Supported by the ADO Data Provider

OLE DB Object	Interface	
ErrorLookup	IErrorLookup	
Command	IAccessor	IConvertType
	IColumnsInfo	IColumnsRowset
	ICommand	ICommandPrepare
	ICommandProperties	ICommandWithParameters
	ICommandText	ISupportErrorInfo
Data Source	IDBCreateSession	IPersist
	IDBInfo	IPersistFile
	IDBInitialize	ISupportErrorInfo
	IDBProperties	
Enumerator	IDBInitialize	
	IDBProperties	
	IParseDisplayName	
	ISourcesRowset	
	ISupportErrorInfo	
MultipleResults	IMultipleResults	
	ISupportErrorInfo	

Table 4-1. Objects and Interfaces Supported by the ADO Data Provider (cont.)

OLE DB Object	Interface	
Rowset	IAccessor	IRowsetIdentity
	IColumnsInfo	IRowsetInfo
	IConvertType	IRowsetLocate
	IColumnsRowset	IRowsetScroll
	IRowset	IRowsetUpdate
	IRowsetChange	ISupportErrorInfo
Session	IDBCreateCommand	ISupportErrorInfo
	IDBSchemaRowset	ITransaction
	IGetDataSource	ITransactionJoin
	IOpenRowset	ITransactionLocal
	ISessionProperties	
Transaction	ITransaction	
	ISupportErrorInfo	
Transaction Options	ITransactionOptions	
	ISupportErrorInfo	

Supported Schema Rowsets

Table 4-2 lists the OLE DB schema rowsets supported by the ADO data provider.

Table 4-2. OLE DB Schema Rowsets Supported by the ADO Data Provider

Schema Call	DB2 UDB	Informix	Oracle	Microsoft SQL Server	Sybase
DBSCHEMA_CATALOGS		X		X	
DBSCHEMA_COLUMN_PRIVILEGES	X	X		X	X
DBSCHEMA_COLUMNS	X	X	X	X	X
DBSCHEMA_FOREIGN_KEYS	X	X	X	X	X
DBSCHEMA_INDEXES	X	X	X	X	X
DBSCHEMA_PRIMARY_KEYS	X	X	X	X	X
DBSCHEMA_PROCEDURE_COLUMNS	X	X		X	X
DBSCHEMA_PROCEDURE_PARAMETERS	X	X	X	X	X
DBSCHEMA_PROCEDURES	X	X	X	X	X
DBSCHEMA_PROVIDER_TYPES	X	X	X	X	X
DBSCHEMA_REFERENTIAL_ CONSTRAINTS		X			
DBSCHEMA_SCHEMATA	X	X		X	X
DBSCHEMA_STATISTICS	X	X	X	X	X
DBSCHEMA_TABLE_CONSTRAINTS		X			
DBSCHEMA_TABLE_PRIVILEGES	X	X		X	X
DBSCHEMA_TABLES	X	X	X	X	X
DBSCHEMA_VIEWS	X	X			X

Supported OLE DB Property Groups

The data provider defines the properties that apply to data sources, and properties that provide a read-only set of information about the data provider and the data source.

To obtain a data provider's property values, a data consumer calls one of the methods listed in [Table 4-3](#).

Table 4-3. OLE DB Property Groups Supported by the ADO Provider

Property Group	Method Used to Obtain Values
Data Source	IDBProperties::GetProperties
Data Source Information	IDBProperties::GetProperties
Initialization	IDBProperties::GetProperties
Rowset	ICommandProperties::GetProperties IRowsetInfo::GetProperties
Session	ISessionProperties::GetProperties

Data Source Property Group

The ADO data provider supports the following property in the DBPROP_DATASOURCE property set. For more information, refer to your Microsoft OLE DB programming documentation.

Table 4-4. OLE DB Data Source Property Supported by the ADO Data Provider

Property Name	Description
DBPROP_CURRENTCATALOG	The name of the current catalog. The data consumer can use the CATALOGS rowset to enumerate catalogs. If unspecified, the data provider uses the default catalog.

Data Source Information Property Group

[Table 4-5](#) lists the properties in the DBPROPSET_DATASOURCEINFO property set supported by the ADO data provider. These properties are in the Data Source Information property group, are read-only properties, and constitute a set of static information about the data provider and data source. For more information about these properties, refer to your Microsoft OLE DB programming documentation.

NOTE: Some values are database-specific and depend on the SequeLink service you are using. These database-specific values are not listed in the table.

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Data Provider

Property ID	Default Value and Description
DBPROP_ACTIVESESSIONS	VALUE=0. There is no limit to the number of sessions that can exist at one time.
DBPROP_ASYNCCTXNABORT	VALUE=VARIANT_FALSE. The data provider cannot abort transactions asynchronously.
DBPROP_ASYNCCTXNCOMMIT	VALUE=VARIANT_FALSE. The data provider cannot commit transactions asynchronously.
DBPROP_BYREFACCESSORS	VALUE=VARIANT_FALSE. The data provider does not support the DBACCESSOR_PASSBYREF flag.
DBPROP_CATALOGLOCATION	The value depends on the SequeLink service you are using.
DBPROP_CATALOGTERM	Specifies the name the data source uses for a catalog. VALUE="Database"
DBPROP_CATALOGUSAGE	A combination of zero or one or more of the following: VALUE=DBPROPVAL_CU_DML_STATEMENTS. Catalog names are supported in all Data Manipulation Language (DML) statements. VALUE=DBPROPVAL_CU_TABLE_DEFINITION. Catalog names are supported in all table definition statements. VALUE=DBPROPVAL_CU_INDEX_DEFINITION. Catalog names are supported in all index definition statements. VALUE= DBPROPVAL_CU_PRIVILEGE_DEFINITION. Catalog names are supported in all privilege definition statements.
DBPROP_COLUMNDEFINITION	VALUE=DBPROPVAL_CD_NOTNULL. Columns can be created non-nullable.

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Data Provider (cont.)

Property ID	Default Value and Description
DBPROP_CONCATNULLBEHAVIOR	The value depends on the SequeLink service you are using.
DBPROP_CONNECTIONSTATUS	VALUE=DBPROPVAL_CS_INITIALIZED. The data source is in an initialized state and able to communicate with the data store.
DBPROP_DATASOURCENAME	Specifies the name of the data source used during connection. The data source is defined using the DataDirect Configuration Manager. See Chapter 3 “Using the ADO Client” on page 127 for more information about configuring ADO data sources.
DBPROP_DATASOURCEREADONLY	VALUE=VARIANT_FALSE. The data source can be updated.
DBPROP_DBMSNAME	Specifies the name of the data store accessed by the ADO provider. The value depends on the SequeLink service you are using.
DBPROP_DBMSVER	Specifies the version of the DBMS that data provider is currently accessing. This value depends on the SequeLink service you are using.
DBPROP_DSOTHREADMODEL	VALUE=DBPROP_RT_FREETHREAD. The ADO provider supports the free-threading model.
DPROP_GROUPBY	This value depends on the SequeLink service you are using.
DBPROP_HETEROGENEOUSTABLES	VALUE=0. Heterogeneous joins are not supported.
DBPROP_IDENTIFIERCASE	This value depends on the SequeLink service you are using.
DBPROP_MAXINDEXSIZE	This value depends on the SequeLink service you are using.
DBPROP_MAXROWSIZE	VALUE=0. There is no limit on the maximum length of a single row in a table.

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Data Provider (cont.)

Property ID	Default Value and Description
DBPROP_MAXROWSIZEINCLUDESBLOB	VALUE=VARIANT_FALSE. The maximum row size does not include the length of all BLOB data.
DBPROP_MAXTABLEINSELECT	VALUE=0. There is no limit on the number of tables allowed in the FROM clause of a Select statement.
DBPROP_MULTIPLEPARAMSETS	VALUE=VARIANT_TRUE. The data provider supports multiple parameter sets at the same time.
DBPROP_MULTIPLERESULTS	VALUE=DBPROPVAL_MR_SUPPORTED. The data provider supports multiple results objects.
DBPROP_MULTIPLESTORAGEOBJECTS	VALUE=VARIANT_TRUE. The data provider supports more than one open storage object at a time.
DBPROP_MULTITABLEUPDATE	VALUE=VARIANT_FALSE. The data provider cannot update rowsets derived from multiple tables.
DBPROP_NULLCOLLATION	The value depends on the SequeLink service you are using.
DBPROP_OLEOBJECTS	VALUE=DBPROPVAL_OO_BLOB. The data provider supports access to BLOBs as structured storage objects.
DBPROP_OPENROWSET SUPPORT	VALUE=DBPROPVAL_OR_S_TABLE. The data provider supports opening tables through IOpenRowset.
DBPROP_ORDERBYCOLUMNSINSELECT	This value depends on the SequeLink service you are using.
DBPROP_OUTPUTPARAMETERAVAILABILITY	VALUE=DBPROPVAL_OA_ATROWRELEASE. If a command returns a single result that is a rowset, output parameter data is available at the time the rowset is completely released.
DBPROP_PERSISTENTIDTYPE	VALUE=DBPROPVAL_PT_NAME. The data provider uses this type of DBID when storing (persisting) DBIDs for tables and columns.

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Data Provider (cont.)

Property ID	Default Value and Description
DBPROP_PREPAREABORTBEHAVIOR	The value depends on the SequeLink service you are using.
DBPROP_PREPARECOMMITBEHAVIOR	The value depends on the SequeLink service you are using.
DBPROP_PROCEDURETERM	This value depends on the SequeLink service you are using.
DBPROP_PROVIDERFRIENDLYNAME	Specifies the name of data provider, for example, SequeLink for ADO Provider.
DBPROP_PROVIDERNAME	VALUE= <code>s1slknn.DLL</code> is the default, where <i>nn</i> is the release level of data provider.
DBPROP_PROVIDEROLEDBVER	Refer to the README file for the supported version of OLE DB supported by the data provider.
DBPROP_PROVIDERVER	Refer to the README file for the supported version.
DBPROP_QUOTEDIDENTIFIERCASE	This value depends on the SequeLink service you are using.
DBPROP_ROWSETCONVERSIONSONCOMMAND	VALUE= <code>VARIANT_TRUE</code> . Callers can inquire on a command about conversions supported on rowsets generated by the command.
DBPROP_SCHEMATERM	This value depends on the SequeLink service you are using.
DBPROP_SCHEMAUSAGE	The value depends on the SequeLink service you are using.
DBPROP_SERVERNAME	Specifies the name of the server.
DBPROP_SQLSUPPORT	This value depends on the SequeLink service you are using.
DBPROP_STRUCTUREDSTORAGE	VALUE= <code>DBPROPVAL_SS_ISEQUENTIALSTREAM</code> .

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Data Provider (cont.)

Property ID	Default Value and Description
DBPROP_SUBQUERIES	<p>A combination of zero, one, or more of the following:</p> <p>VALUE=DBPROPVAL_SQ_COMPARISON</p> <p>VALUE=</p> <p>DBPROPVAL_SQ_CORRELATEDSUBQUERIES.</p> <p>This indicates that all predicates that support subqueries support correlated subqueries.</p> <p>VALUE=DBPROPVAL_SQ_EXISTS</p> <p>VALUE=DBPROPVAL_SQ_IN</p> <p>VALUE=DBPROPVAL_SQ_QUANTIFIED</p>
DBPROP_SUPPORTEDTXNDDL	This value depends on the SequeLink service you are using.
DBPROP_SUPPORTEDTXNISOLEVELS	This value depends on the SequeLink service you are using.
DBPROP_SUPPORTEDTXNISORETAIN	VALUE=0. The data provider supports no transaction isolation retention levels.
DBPROP_TABLETERM	<p>Specifies the name the data source uses for a table.</p> <p>VALUE="Table"</p>
DBPROP_USERNAME	Specifies a character string with the name used in a particular database. This can be different from the login name.

Initialization Property Group

Table 4-6 provides the supported initialization properties for the ADO provider. The properties are read/write. For more information, refer to the Microsoft OLE DB programming documentation.

Table 4-6. Initialization Properties Supported by the ADO Data Provider

Property Name	Default Value and Description
DBPROP_AUTH_PASSWORD	<p>Specifies the password to be used for connecting to the data source or enumerator.</p> <p>This corresponds to the Password connection attribute. See “Connecting with a Provider String” on page 155 for more information about connection attributes.</p>
DBPROP_AUTH_PERSIST_SENSITIVE_AUTHINFO	<p>VALUE=VARIANT_FALSE. The data source object cannot store the password or other sensitive authentication information.</p>
DBPROP_AUTH_USERID	<p>Specifies the User ID to be used for connecting to the data source.</p> <p>This corresponds to the User ID connection attribute. See “Connecting with a Provider String” on page 155 for more information about connection attributes,.</p>
DBPROP_INIT_CATALOG	<p>Specifies the name of the initial or default catalog to use when connecting to the data source.</p>
DBPROP_INIT_DATASOURCE	<p>Specifies the name of the data source or enumerator to which to connect.</p>
DBPROP_INIT_HWND	<p>Specifies the window handle to use if the data source object or enumerator needs to prompt for additional information.</p>
DBPROP_INIT_LCID	<p>Specifies the preferred locale ID for the consumer.</p>

Table 4-6. Initialization Properties Supported by the ADO Data Provider (cont.)

Property Name	Default Value and Description
DBPROP_INIT_MODE	VALUE=DB_MODE_READ. The default is read-only.
DBPROP_INIT_OLEDBSERVICES	VALUE=0. The data provider does not enable the OLE DB services.
DBPROP_INIT_PROMPT	VALUE=DBPROMPT_COMPLETE. Display a connection dialog only if missing information is needed.
DBPROP_INIT_PROVIDERSTRING	Specifies a provider-specific string that contains extra initialization information. See "Connecting with a Provider String" on page 155 for information about using the provider string. Consumers should use this property only for provider-specific connection information as described in Table 3-2 on page 157 .

Rowset Property Group

OLE DB data consumers can request certain properties to be satisfied by the rowsets that result from an OpenRowset or ICommand::Execute call. Common properties include the set of interfaces to be supported by the resulting rowset.

The ADO data provider supports the IRowsetIdentity interface. This allows the data consumer to determine when two row handles represent the same underlying data.

Table 4-7 provides the properties that are included in the ADO provider properties group. The table shows the initial default values. Some of these properties can be changed at the Command level (through ICommandProperties->SetProperties) or Session level (through IOpenRowset). The resulting rowset will contain different values for these properties.

Table 4-7. Rowset Properties Supported by the ADO Data Provider

Property Name	Default Value and Description
DBPROP_ABORTPRESERVE	The value depends on the SequeLink service you are using.
DBPROP_ACCESSORDER	VALUE= DBPROPVAL_AO_SEQUENTIALSTORAGEOBJECTS Columns bound as storage objects can only be accessed in sequential order as determined by the column ordinal.
DBPROP_BLOCKINGSTORAGEOBJECTS	VALUE=VARIANT_FALSE. Instantiated storage objects do not prevent the use of other methods.
DBPROP_BOOKMARKINFO	VALUE=DBPROPVAL_BI_CROSSROWSET (if bookmarks are supported) VALUE=0 (if bookmarks are not supported)
DBPROP_BOOKMARKS	VALUE=FALSE. The rowset does not support bookmarks.
DBPROP_BOOKMARKSKIPPED	VALUE=VARIANT_FALSE. GetRowsAt, GetApproximatePosition, or FindNextRow returns DB_E_BADBOOKMARK.
DBPROP_BOOKMARKTYPE	VALUE=DBPROPVAL_BMK_NUMERIC. The bookmark type is numeric.
DBPROP_CANFETCHBACKWARDS	VALUE=VARIANT_FALSE. cRows must be non-negative.
DBPROP_CANHOLDROWS	VALUE=VARIANT_FALSE. The rowset might require pending changes to be transmitted to the data store before fetching additional rows.
DBPROP_CANSROLLBACKWARDS	VALUE=VARIANT_FALSE. IRowsOffset must be non-negative.

Table 4-7. Rowset Properties Supported by the ADO Data Provider (cont.)

Property Name	Default Value and Description
DBPROP_CHANGEINSERTEDROWS	VALUE=VARIANT_FALSE. DeleteRows returns a status of DBROWSTATUS_E_NEWLYINSERTED for newly inserted rows, and SetData returns DB_E_NEWLYINSERTED.
DBPROP_COLUMNRESTRICT	Specifies whether access rights are restricted on a column-by-column basis. VALUE=VARIANT_TRUE. Access rights are restricted on a column-by-column basis. VALUE=VARIANT_FALSE. Access rights are not restricted on a column-by-column basis.
DBPROP_COMMITPRESERVE	The value is specific to the SequeLink Server you are using.
DBPROP_DELAYSTORAGEOBJECTS	VALUE=VARIANT_FALSE. Storage objects are used in immediate update mode.
DBPROP_IAccessor	VALUE=VARIANT_TRUE
DBPROP_IColumnsInfo	VALUE=VARIANT_TRUE
DBPROP_IColumnsRowset	VALUE=VARIANT_TRUE
DBPROP_IConvertType	VALUE=VARIANT_TRUE
DBPROP_IMultipleResults	VALUE=VARIANT_FALSE
DBPROP_IRowset	VALUE=VARIANT_TRUE
DBPROP_IRowsetChange	VALUE=VARIANT_TRUE
DBPROP_IRowsetIdentity	VALUE=VARIANT_FALSE
DBPROP_IRowsetInfo	VALUE=VARIANT_TRUE
DBPROP_IRowsetLocate	VALUE=VARIANT_TRUE
DBPROP_IRowsetRefresh	VALUE=VARIANT_FALSE
DBPROP_IRowsetScroll	VALUE=VARIANT_FALSE
DBPROP_IRowsetUpdate	VALUE=VARIANT_FALSE
DBPROP_ISequentialStream	VALUE=VARIANT_FALSE
DBPROP_ISupportErrorInfo	VALUE=VARIANT_TRUE
DBPROP_IMMOBILEROWS	VALUE=VARIANT_TRUE. The rowset will not reorder inserted or updated rows.

Table 4-7. Rowset Properties Supported by the ADO Data Provider *(cont.)*

Property Name	Default Value and Description
DBPROP_LITERALBOOKMARKS	VALUE=VARIANT_FALSE. Bookmarks can only be compared with IRowsetLocate::Compare.
DBPROP_LITERALIDENTITY	VALUE=VARIANT_FALSE. The consumer must call IRowsetIdentity::IsSameRow to determine whether two row handles point to the same row.
DBPROP_LOCKMODE	VALUE=DBPROPVAL_LM_NONE. The data provider is not required to lock rows to ensure successful updates.
DBPROP_MAXOPENROWS	Specifies the maximum number of rows that can be active at the same time. VALUE=4096
DBPROP_MAXPENDINGROWS	VALUE=0. There is no limit on the number of rows that can have pending changes at the same time.
DBPROP_MAXROWS	VALUE=0. There is no limit on the number of rows that can be returned in a rowset.
DBPROP_MEMORYUSAGE	VALUE=0. There is no limit on the amount of memory that the rowset can use.
DBPROP_OTHERINSERT	VALUE=VARIANT_FALSE. The rowset cannot see updates and deletes made by others.
DBPROP_OTHERUPDATEDELETE	VALUE=VARIANT_FALSE. The rowset cannot see updates and deletes made by others.
DBPROP_OWNINSERT	VALUE=VARIANT_FALSE. The rowset cannot see rows inserted by consumers of the rowset unless the command is executed again.
DBPROP_OWNUPDATEDELETE	VALUE=VARIANT_FALSE. The rowset cannot see updates and deletes made by consumers of the rowset unless the command is executed again.
DBPROP_QUICKRESTART	VALUE=VARIANT_TRUE. IRowset::RestartPosition is not expensive to execute and does not execute the command that created the rowset again.
DBPROP_REENTRANTEVENTS	VALUE=VARIANT_TRUE. The data provider supports reentrancy during callbacks to the IRowsetNotify interface.

Table 4-7. Rowset Properties Supported by the ADO Data Provider (cont.)

Property Name	Default Value and Description
DBPROP_REMOVEDELETED	VALUE=VARIANT_FALSE. Static cursors do not remove deleted rows.
DBPROP_REPORTMULTIPLECHANGES	VALUE=VARIANT_FALSE. An update or delete always affects a single row or data provider cannot detect whether it affects multiple rows.
DBPROP_RETURNPENDINGINSERTS	VALUE=VARIANT_FALSE. The methods that fetch rows cannot return pending insert rows.
DBPROP_ROWRESTRICT	VALUE=VARIANT_FALSE. Access rights are not restricted on a row-by-row basis.
DBPROP_ROWTHREADMODEL	VALUE=DBPROPVAL_RT_FREETHREAD. The data provider uses the free-threaded model.
DBPROP_SERVERCURSOR	VALUE=VARIANT_FALSE. The data provider determines where to locate the cursor.
DBPROP_STRONGIDENTITY	VALUE=VARIANT_FALSE. There is no guarantee that the handles of newly inserted rows can be compared successfully.
DBPROP_TRANSACTEDOBJECT	VALUE=VARIANT_FALSE. Any object created on the specified column is not transacted.
DBPROP_UNIQUEROWS	VALUE=VARIANT_FALSE. Rows in the rowset may or may not be uniquely identified by their column values.
DBPROP_UPDATABILITY	<p>A combination of zero or one or more of the following:</p> <p>Value=DBPROPVAL_UP_CHANGE. SetData is supported.</p> <p>Value=DBPROPVAL_UP_DELETE. DeleteRows is supported.</p> <p>Value=DBPROPVAL_UP_INSERT. InsertRow is supported.</p>

Session Property Group

[Table 4-8](#) lists the properties the ADO data provider supports in the DBPROPSET_SESSION property set. For more information, refer to your Microsoft OLE DB programming information.

Table 4-8. Session Properties Supported by the ADO Data Provider

Property Name	Default Value and Description
DBPROP_SESS_ AUTOCOMMITISOLEVELS	Specifies the transaction isolation level while in auto-commit mode. VALUE=NULL

OLE DB Interfaces Supported in ADO

[Table 4-9](#) lists the OLE DB interfaces that ADO supports and describes whether the interface is required by ADO. The ADO data provider supports additional OLE DB interfaces that are not used by ADO.

See [“OLE DB Objects and Interfaces” on page 172](#) for a description of the OLE DB objects and interfaces supported by the ADO data provider.

Table 4-9. Supported OLE DB Interfaces Used by ADO

OLE DB Interface	Use by ADO
IAccessor	Required
IColumnsInfo	Required
IColumnsRowset	If available
ICommand	If available

Table 4-9. Supported OLE DB Interfaces Used by ADO *(cont.)*

OLE DB Interface	Use by ADO
ICommandPrepare	If available
ICommandProperties	If available
ICommandText	If available
ICommandWithParameters	If available
IConvertType	Required
IDBCreateCommand	If available
IDBCreateSession	Required
IDBInitialize	Required
IDBProperties	Required
IOpenRowset	Required
IRowset	Required
IRowsetChange	If available
IRowsetIndex	If available
IRowsetInfo	Required
IRowsetLocate	If available
IRowsetScroll	If available
IRowsetUpdate	If available
ISourcesRowset	If available
ITransaction	If available
ITransactionLocal	If available
ITransactionOptions	If available

Mapping ADO Methods and Properties

This section maps the methods, properties, and collections of ADO objects to the OLE DB methods supported by the ADO data provider.

Some ADO methods do not have a comparable OLE DB method. When more than one OLE DB method can be used, ADO uses the method that requires the least amount of system resources. For example, if an ADO method can use either ICommand or IOpenRowset, it uses the less performance-intensive IOpenRowset.

ADO Command Object

The Command object can be used to specify a database query in the language native to the database server. For a relational data provider, this is usually a SQL statement.

The Execute method for the ADO Command object maps to the OLE DB method, ICommand::Execute.

[Table 4-10](#) lists the dynamic properties that are supported by the ADO data provider for the Command object.

Table 4-10. Dynamic Properties Used for the ADO Command Object

ADO Property	Default Value and Description
Access Order	VALUE=2. Columns can be accessed in any order.
Blocking Storage Objects	VALUE=False. Instantiated storage objects do not prevent the use of other methods.
Change Inserted Rows	VALUE=False. The value can only be set to True if the rowset is using a keyset-driven cursor.

Table 4-10. Dynamic Properties Used for the ADO Command Object (cont.)

ADO Property	Default Value and Description
Column Privileges	Specifies whether access rights are restricted on a column-by-column basis. VALUE=True. Access rights are restricted on a column-by-column basis. VALUE=False. Access rights are not restricted on a column-by-column basis. If the rowset exposes IRowsetChange, SetData can be called for any column in the rowset.
Fetch Backwards	VALUE=False. cRows must be non-negative.
Hold Rows	VALUE=True. Access rights are restricted on a column-by-column basis.
IAccessor	VALUE=True
IColumnsInfo	VALUE=True
IColumnsRowset	VALUE=True
IConvertType	VALUE=True
IRowset	VALUE=True
IRowsetChange	VALUE=False
IRowsetInfo	VALUE=True
Literal Row Identity	VALUE=False. The consumer must call IRowsetIdentity::IsSameRow to determine whether two row handles point to the same row.
Lock Mode	VALUE=1. The data provider is not required to lock rows at any time to ensure successful updates.
Maximum Open Rows	Specifies the maximum number of rows that can be active at the same time. VALUE=4096
Maximum Pending Rows	VALUE=0. There is no limit on the number of rows that can have pending changes at the same time.
Maximum Rows	VALUE=0. There is no limit on the number of rows that can be returned in a rowset.
Memory Usage	VALUE=0. There is no limit on the amount of memory that can be used by the rowset.

Table 4-10. Dynamic Properties Used for the ADO Command Object (cont.)

ADO Property	Default Value and Description
Objects Transacted	VALUE=True. Any object created on the specified column is transacted.
Others' Changes Visible	VALUE=False. The rowset cannot see updates and deletes made by others.
Others' Inserts Visible	VALUE=False. The rowset cannot see inserts made by others.
Own Changes Visible	VALUE=False. The rowset cannot see updates and deletes made by consumers of the rowset unless the command is executed again.
Own Inserts Visible	VALUE=False. The rowset can see the rows inserted by consumers only after the command is run again.
Preserve on Abort	The value depends on the SequeLink service that you are using.
Preserve on Commit	The value depends on the SequeLink service that you are using.
Quick Restart	VALUE=True. IRowset::RestartPosition is relatively quick to execute. It does not again execute the command that created the rowset.
Remove Deleted Rows	VALUE=False. Static cursors do not remove deleted rows.
Report Multiple Changes	VALUE=False. An update or delete always affects a single row or the data provider cannot detect whether it affects multiple rows.
Return Pending Inserts	VALUE=False. The methods that fetch rows cannot return pending insert rows.
Row Privileges	VALUE=False. The data provider does not set access restrictions for rows.
Row Threading Model	VALUE=1. The ADO data provider uses the free-threaded model.
Scroll Backward	VALUE=False. IRowsOffset must be non-negative.
Server Cursor	VALUE=False. The data provider determines where to locate the cursor.
Strong Row Identity	VALUE=False. There is no guarantee that the handles of newly inserted rows can be compared successfully.

Table 4-10. Dynamic Properties Used for the ADO Command Object (cont.)

ADO Property	Default Value and Description
Unique Rows	VALUE=False. Rows in the rowset may or may not be uniquely identified by their column values.
Updatability	Specifies the supported methods on IRowsetChange. VALUE=0
Use Bookmarks	VALUE=False. The rowset does not support bookmarks.

Connection Object

The ADO Connection object represents a single session with an OLE DB data source. It defines a physical connection to the data source for a data provider.

[Table 4-11](#) lists the supported ADO methods for the Connection object and maps them to the corresponding OLE DB methods.

Table 4-11. Mapping Methods Supported by the ADO Connection Object

ADO Method	OLE DB Method
BeginTrans	ITransactionLocal::StartTransaction
CommitTrans	ITransactionLocal::Commit
Execute	ICommand::Execute or IOpenRowset::OpenRowset
Open	IDBInitialize::Initialize IDBCreateSession::Create Session
RollBackTrans	ITransactionLocal::Abort
OpenSchema	IDBSchemaRowset::GetRowset

Table 4-12 lists the dynamic properties supported for the ADO Connection object.

Table 4-12. Dynamic Properties Supported for the ADO Connection Object

ADO Property	Default Value and Description
Active Sessions	VALUE=0. There is no limit to the maximum number of sessions that can exist at one time.
Asynchable Abort	VALUE=False. The data provider cannot abort transactions asynchronously.
Asynchable Commit	VALUE=False. The data provider cannot commit transactions asynchronously.
Autocommit Isolation Levels	<p>Specifies the transaction isolation level while in auto-commit mode. A combination of zero or one or more of the following:</p> <p>VALUE=DBPROPVAL_TI_BROWSE</p> <p>VALUE=DBPROPVAL_TI_CURSORSTABILITY</p> <p>VALUE=DBPROPVAL_TI_ISOLATED</p> <p>VALUE=DBPROPVAL_TI_READCOMMITTED</p> <p>VALUE=DBPROPVAL_TI_READUNCOMMITTED</p> <p>VALUE=DBPROPVAL_TI_REPEATABLEREAD</p> <p>VALUE=DBPROPVAL_TI_SERIALIZABLE</p>
Catalog Location	<p>The value depends on the SequeLink service you are using. Possible values are:</p> <p>VALUE=1. The catalog name is at the start of the fully qualified name.</p> <p>VALUE=2. The catalog name is at the end of the fully qualified name.</p>
Catalog Term	<p>Specifies the name the data source uses for a catalog.</p> <p>VALUE=Database</p>

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Catalog Usage	<p>Specifies how catalog names can be used in text commands. A combination of zero or one or more of the following:</p> <p>VALUE=1. Catalog names are supported in all Data Manipulation Language statements.</p> <p>VALUE=2. Catalog names are supported in all table definition statements.</p> <p>VALUE=4. Catalog names are supported in all index definition statements.</p> <p>VALUE=8. Catalog names are supported in all privilege definition statements.</p>
Column Definition	VALUE=1. Columns can be created non-nullable.
COM Object Support	VALUE=1. The data providers support access to BLOBs as structured storage objects. A data consumer determines which interfaces are supported through the Structured Storage property.
Connection Status	VALUE=1. The data source is in an initialized state and able to communicate with the data store.
Current Catalog	Specifies the name of the current catalog. The data consumer can use the CATALOGS rowset to enumerate catalogs. If not set, the data provider uses the default catalog.
Data Source	Specifies the name of the data source or enumerator to which to connect.
Data Source Name	Specifies the name of the data source (server) used during the connection process.
Data Source Object Threading Model	VALUE=1. The data provider uses the free-threading model.
DBMS Name	Specifies the name of the product accessed by the ADO data provider.
DBMS Version	Specifies the version of the product that the data provider is currently accessing.

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Extended Properties	<p>A provider-specific string that contains extra initialization information. Consumers should use this property only for provider-specific connection information.</p> <p>See “Connecting with a Provider String” on page 155 for information on using the provider string with the ADO data provider.</p>
GROUP BY Support	<p>This value depends on the SequeLink service you are using.</p> <p>VALUE=1. The GROUP BY clause is not supported.</p> <p>VALUE=2. The GROUP BY clause must contain all nonaggregated columns in the select list. It cannot contain any other columns.</p> <p>VALUE=4. The GROUP BY clause must contain all nonaggregated columns in the select list. It can contain columns that are not in the select list.</p> <p>VALUE=8. The columns in the GROUP BY clause and the select list are not related. The meaning of nongrouped, nonaggregated columns in the select list is data source-dependent.</p>
Heterogeneous Table Support	<p>VALUE=0. The data provider cannot join tables from different catalogs or providers.</p>
Identifier Case Sensitivity	<p>This value depends on the SequeLink service you are using.</p> <p>VALUE=1. Identifiers in SQL are case-sensitive and are stored in upper case in the system catalog.</p> <p>VALUE=2. Identifiers in SQL are case-insensitive and are stored in lower case in the system catalog.</p> <p>VALUE=4. Identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog.</p> <p>VALUE=8. Identifiers in SQL are case-insensitive and are stored in mixed case in the system catalog.</p>
Initial Catalog	<p>Specifies the name of the initial, or default, catalog to use when connecting to the data source.</p>

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Isolation Levels	<p>This value depends on the SequeLink service you are using. Zero, or a combination of one or more of the following:</p> <p>VALUE=DBPROPVAL_TI_BROWSE</p> <p>VALUE=DBPROPVAL_TI_CHAOS</p> <p>VALUE=DBPROPVAL_TI_CURSORSTABILITY</p> <p>VALUE=DBPROPVAL_TI_ISOLATED</p> <p>VALUE=DBPROPVAL_TI_READCOMMITTED</p> <p>VALUE=DBPROPVAL_TI_READUNCOMMITTED</p> <p>VALUE=DBPROPVAL_TI_REPEATABLEREAD</p> <p>VALUE=DBPROPVAL_TI_SERIALIZABLE</p>
Isolation Retention	VALUE=0. The data provider supports no transaction isolation retention levels.
Locale Identifier	Specifies the preferred locale ID for the consumer.
Maximum Index Size	VALUE=0. There is no limit on the index size.
Maximum Open Chapters	VALUE=0. There is no limit on the maximum number of chapters that can be open at any time.
Maximum Row Size	VALUE=0. There is no limit on the maximum length of a single row in a table.
Maximum Row Size Includes BLOB	VALUE=False. The maximum row size does not include the length of all BLOB data.
Maximum Tables in SELECT	VALUE=0. There is no limit on the number of tables in a Select statement.
Mode	VALUE=3. The default access is read-write.
Multi-Table Update	VALUE=False. The data provider cannot update rowsets derived from multiple tables.
Multiple Connections	VALUE=True. The data provider must spawn multiple connections to support concurrent command, session, and rowset objects.
Multiple Parameter Sets	VALUE=True. The data provider supports multiple parameter sets at the same time.

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Multiple Results	VALUE=1. The data provider supports multiple results objects.
Multiple Storage Objects	VALUE=True. The data provider supports more than one open storage object at a time.
NULL Collation Order	The value depends on the SequeLink service you are using.
NULL Concatenation Behavior	The value depends on the SequeLink service you are using.
OLE DB Services	VALUE=0. The data provider does not enable the OLE DB services.
OLE DB Version	Specifies the version of OLE DB supported by the data provider. Refer to the README for the supported version.
Open Rowset Support	VALUE=0. All data providers support opening tables through IOpenRowset.
ORDER BY Columns In Select List	This value depends on the SequeLink service you are using.
Output Parameter Availability	VALUE=4. If a command returns a single result that is a rowset, output parameter data is available at the time the rowset is completely released.
Pass By Ref Accessors	VALUE=False. The data provider does not support the DBACCESSOR_PASSBYREF flag.
Password	Specifies the password to be used for connecting to the data source or enumerator.
Prepare Abort Behavior	The value depends on the SequeLink service you are using.
Prepare Commit Behavior	The value depends on the SequeLink service you are using.
Procedure Term	Specifies a character string that contains the data source vendor's name for a procedure.
Prompt	Specifies whether to prompt the user for additional information during initialization.
Provider Friendly Name	Specifies the name of the data provider, DataDirect SequeLink for ADO Provider.

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Provider Name	VALUE=slslknn.DLL, where <i>nn</i> is the release level of the data provider.
Provider Version	Specifies the version of the DataDirect data provider. Refer to the README for the version number.
Quoted Identifier Sensitivity	<p>This value depends on the SequeLink service you are using.</p> <p>VALUE=1. Quoted identifiers in SQL are case-sensitive and are stored in upper case in the system catalog.</p> <p>VALUE=2. Quoted identifiers in SQL are case-insensitive and are stored in lower case in the system catalog.</p> <p>VALUE=4. Quoted identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog.</p> <p>VALUE=8. Quoted identifiers in SQL are case-insensitive and are stored in mixed case.</p>
Read Only Data Source	VALUE=0. The data source can be updated.
Rowset Conversions on Command	VALUE=True. Callers can inquire on a command about conversions supported on rowsets generated by the command.
Schema Term	Specifies the name the data source uses for a schema.
Schema Usage	<p>This value depends on the SequeLink service you are using. A combination of the following:</p> <p>VALUE=DBPROPVAL_SU_DML_STATEMENTS. Schema names are supported in all Data Manipulation Language statements.</p> <p>VALUE=DBPROPVAL_SU_TABLE_DEFINITION. Schema names are supported in all table definition statements.</p> <p>VALUE=DBPROPVAL_SU_INDEX_DEFINITION. Schema names are supported in all index definition statements.</p> <p>VALUE=DBPROPVAL_SU_PRIVILEGE_DEFINITION. Schema names are supported in all privilege definition statements.</p>

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Server Name	Specifies the name of the server. This can be the same as the Data Source property if the server name is used to define the user's data source. Alternatively, if the data provider connects through the data provider's "friendly" data source names, this can be the actual name of the server.
SQL Support	<p>Specifies the level of SQL grammar that the data provider supports. The effects are cumulative. A combination of zero or one or more of the following:</p> <p>VALUE=DBPROPVAL_SQL_NONE</p> <p>VALUE=DBPROPVAL_SQL_ODBC_CORE</p> <p>VALUE=DBPROPVAL_SQL_ODBC_MINIMUM</p> <p>VALUE=DBPROPVAL_SQL_ODBC_EXTENDED</p> <p>VALUE=DBPROPVAL_SQL_ESCAPECLAUSES</p> <p>VALUE=DBPROPVAL_SQL_ANSI92_ENTRY</p> <p>VALUE=DBPROPVAL_SQL_ANSI92_ENTRY</p> <p>VALUE=DBPROPVAL_SQL_FIPS_TRANSITIONAL</p> <p>VALUE=DBPROPVAL_SQL_ANSI92_INTERMEDIATE</p> <p>VALUE=DBPROPVAL_SQL_ANSI92_FULL</p> <p>VALUE=DBPROPVAL_SQL_ANSI89_IEF</p> <p>VALUE=DBPROPVAL_SQL_SUBMINIMUM</p>
Structured Storage	VALUE=1. The data provider supports DBPROPVAL_SS_ISEQUENTIALSTREAM.
Subquery Support	<p>Specifies the predicates in text commands that support subqueries. The value can be zero, or one or more of the following:</p> <p>VALUE=DBPROPVAL_SQ_CORRELATEDSUBQUERIES</p> <p>VALUE=DBPROPVAL_SQ_COMPARISON</p> <p>VALUE=DBPROPVAL_SQ_EXISTS</p> <p>VALUE=DBPROPVAL_SQ_IN</p> <p>VALUE=DBPROPVAL_SQ_QUANTIFIED</p>
Table Term	Specifies the name the data source uses for a table.

Table 4-12. Dynamic Properties Supported for the ADO Connection Object (cont.)

ADO Property	Default Value and Description
Transaction DDL	This value depends on the SequeLink service you are using.
User Name	Specifies a character string with the name used in a particular database. This can be different from the login name.
Window Handle	Specifies the window handle to use if the data source object or enumerator needs to prompt for additional information.

Recordset Object

The Recordset object is the set of records resulting from a query against a database and a cursor, which is the interface to the records. If you create a Connection object before you open a Recordset object, multiple Recordset objects can be opened on the same connection.

[Table 4-13](#) maps the methods of the Recordset object to the OLE DB methods supported by the ADO data provider.

Table 4-13. Mapping Methods Supported by the Recordset Object

ADO Method	OLE DB Method
AddNew	IRowsetChange::InsertRow
CancelBatch	IRowsetUpdate::Undo
Clone	IRowsetLocate
Close	IAccessor::ReleaseAccessor IRowset::ReleaseRows
Delete	IRowsetChange::DeleteRows

Table 4-13. Mapping Methods Supported by the Recordset Object *(cont.)*

ADO Method	OLE DB Method
GetRows	IAccessor::CreateAccessor IRowsetLocate::GetRowsAt IRowset::GetNextRows IRowset::GetData
Move	IRowsetLocate::GetRowsAt IRowset::GetNextRows
MoveFirst	IRowsetLocate::GetRowsAt IRowset::RestartPosition
MoveLast	IRowsetLocate::GetRowsAt
MoveNext	IRowsetLocate::GetRowsAt IRowset::GetNextRows
MovePrevious	IRowsetLocate::GetRowsAt IRowset::GetNextRows
NextRecordSet	IMultipleResults::GetResult
Open	IOpenRowset::OpenRowset ICommand::Execute
Requery	IOpenRowset::OpenRowset ICommand::Execute
Supports	IRowsetInfo::GetProperties
Update	IRowsetChange::SetData IRowsetUpdate::Update
UpdateBatch	IRowsetUpdate::Update

Table 4-14 lists the dynamic properties supported by the ADO data provider for the Recordset object.

Table 4-14. Dynamic Properties Used for the Recordset Object

ADO Property	Default Value and Description
Access Order	VALUE=2. Columns can be accessed in any order.
Blocking Storage Objects	VALUE=False. Instantiated storage objects do not prevent the use of other methods.
Bookmark Information	Specifies additional information about bookmarks over the rowset.
Bookmark Type	VALUE=1. The bookmark type is numeric.
Change Inserted Rows	VALUE=False. DeleteRows returns a status of DBROWSTATUS_E_NEWLYINSERTED for the newly inserted row and SetData returns DB_E_NEWLYINSERTED.
Column Privileges	Specifies whether access rights are restricted on a column-by-column basis. VALUE=True. Access rights are restricted on a column-by-column basis. VALUE=False. Access rights are not restricted on a column-by-column basis. If the rowset exposes IRowsetChange, SetData can be called for any column in the rowset.
Delay Storage Object Updates	VALUE=False. Storage objects are used in immediate update mode.
Fetch Backwards	VALUE=False. cRows must be non-negative.
Hold Rows	VALUE=False. The rowset might require pending changes to be transmitted to the data store before fetching additional rows.
IAccessor	VALUE=True.
IColumnsInfo	VALUE=True.
IColumnsRowset	VALUE=True.
IConvertType	VALUE=True.
IRowset	VALUE=True.
IRowsetChange	VALUE=False.

Table 4-14. Dynamic Properties Used for the Recordset Object (cont.)

ADO Property	Default Value and Description
IRowsetInfo	VALUE=True.
IRowsetLocate	VALUE=False.
Immobile Rows	VALUE=True. The rowset will not reorder inserted or updated rows.
Literal Bookmarks	VALUE=False. Bookmarks can only be compared with IRowsetLocate::Compare.
Literal Row Identity	VALUE=False. The consumer must call IRowsetIdentity::IsSameRow to determine whether two row handles point to the same row.
Lock Mode	VALUE=1. The data provider is not required to lock rows to ensure successful updates.
Maximum Open Rows	Specifies the maximum number of rows that can be active at the same time. VALUE=4096
Maximum Pending Rows	VALUE=0. There is no limit to the maximum number of rows that can have pending changes at the same time.
Maximum Rows	VALUE=0. There is no limit to the number of rows that can be returned in a rowset.
Memory Usage	VALUE=0. There is no limit to the amount of memory that the rowset can use.
Objects Transacted	VALUE=True. Any object created on the specified column is transacted.
Others' Changes Visible	VALUE=False. The rowset cannot see updates and deletes made by others.
Others' Inserts Visible	VALUE=False. The rowset cannot see the rows inserted by others.
Own Changes Visible	VALUE=False. The rowset cannot see updates and deletes made by consumers of the rowset unless the command is executed again.
Own Inserts Visible	VALUE=False. The rowset can see the rows inserted by consumers only after the command is run again.

Table 4-14. Dynamic Properties Used for the Recordset Object (cont.)

ADO Property	Default Value and Description
Preserve on Abort	The value is specific to the SequeLink Server you are using.
Preserve on Commit	The value is specific to the SequeLink Server you are using.
Quick Restart	VALUE=True. IRowset::RestartPosition is not expensive to execute and does not execute the command that created the rowset again.
Remove Deleted Rows	VALUE=False. Static cursors do not remove deleted rows.
Report Multiple Changes	VALUE=False. An update or delete always affects a single row or the data provider cannot detect whether it affects multiple rows.
Return Pending Inserts	VALUE=False. The methods that fetch rows cannot return pending insert rows.
Row Privileges	VALUE=False. Access rights are not restricted on a row-by-row basis.
Row Threading Model	VALUE=1. The data provider supports the free-threaded model.
Scroll Backward	VALUE=False. IRowsOffset must be non-negative.
Server Cursor	VALUE=False. The data provider determines where to locate the cursor.
Skip Deleted Bookmarks	VALUE=False. GetRowsAt, GetApproximatePosition, or FindNextRow returns DB_E_BADBOOKMARK.
Strong Row Identity	VALUE=False. There is no guarantee that the handles of newly inserted rows can be compared successfully.
Unique Rows	VALUE=False. Rows in the rowset may or may not be uniquely identified by their column values.
Updatability	Specifies the supported methods on IRowsetChange. VALUE=0
Use Bookmarks	VALUE=False. The rowset does not support bookmarks.

Data Shaping

Data shaping allows you to create hierarchical recordsets with data exposed by an ADO/OLE DB data provider. This is done through the MSDataShape OLE DB provider, which is part of the MDAC. MSDataShape acts as a service component to the ADO provider to expose data shaping functionality.

To perform queries, MSDataShape uses a Shape language, which is functionally similar to SQL. For more information about the Shape language, refer to your MDAC documentation.

You specify the data provider in the Connection object connect string by typing `Data Provider=DataDirect SequeLink for ADO Provider`. The data provider supplying data shaping support is specified in the Connection object Provider property as `MSDataShape`.

For example, the following code fragment can be used to create hierarchical recordsets with data exposed by the ADO provider using the ADO data source named *HR*:

```
Dim cnn As New ADODB.Connection
cnn.Provider = "MSDataShape"
cnn.Open
"Shape Provider = DataDirect SequeLink for ADO Provider;
DataSourceName = HR;
User ID = Mary Smith;
Password = human"
```

Persisting Information

A data source object can be persisted (saved). The ADO provider uses the `IPersist` and the `IPersistFile` interfaces to persist the class ID and the values of data source properties set by the data consumer. With the `IPersistFile` interface, the data consumer saves the information to a file.

When the data consumer loads the persisted data source, the data provider retrieves the saved information. All of the initialization properties return to the state that was current when the data source was persisted. The stored values overwrite the values of any properties the data consumer might have set.

Using Rowsets

ADO/OLE DB data providers use rowsets to expose data in tabular form. The ADO data provider supports the `IOpenRowset` interface, which retrieves all data from a table for a consumer. In addition, the data provider supports the `ICommand` interface, which allows a consumer to get a rowset that meets a specific criteria.

See the [“Supported Schema Rowsets” on page 174](#) for information on the schema rowsets supported.

For more information about rowsets, refer to your Microsoft OLE DB programming documentation.

Mapping Data Types

See [Appendix B “Data Types and Isolation Levels” on page 451](#) for information on the way the underlying data provider’s data types map to the standard OLE DB data types.

`IColumns::GetColumnInfo` and `ICommandWithParameters::GetParameterInfo` are used to report OLE DB data types.

NOTE: Always use four-digit years for conversions from variant types to date/time types. Using two-digit years is not supported and will result in undefined behavior.

Specifying Application IDs

Application IDs are alphanumeric strings passed by a SequeLink Client that identify the client application to a SequeLink service that has been configured to accept connections only from specific application IDs.

For more information about configuring SequeLink services to accept connections only from specific application IDs, refer to the *SequeLink Administrator’s Guide*.

Specifying Application IDs Explicitly

Using the ADO Client, the client application specifies the following *key-value* pair in the `DBPROP_INIT_PROVIDERSTRING` property of the `DBPROPSET_DBINITALL` property set:

```
ApplicationID=MyAppID;
```

where *myAppID* is the application ID.

Generating Application IDs Automatically

Using the ADO Client, the client application specifies the following *key-value* pairs in the DBPROP_INIT_PROVIDERSTRING property of the DBPROPSET_DBINITALL property set:

Automatic Application ID=*x*

where:

- When Automatic Application ID is set to 1, the full path of the application executable is used as input for the hash function.
- When Automatic Application ID is set to 2, the executable binary file is used as input for the hash function.
- When Automatic Application ID is set to 3, both the full path of the application executable and the executable binary file are used as input for the hash function.
- When Automatic Application ID is set to 4, the full directory name of the application executable is used as input for the hash function.

Error Handling

The following types of errors can occur when you are using the SequeLink *for* ADO Client:

- SequeLink *for* ADO data provider errors
- SequeLink Client errors
- SequeLink Server errors
- Database errors

SequeLink® *for* ADO Provider Errors

An error generated by the SequeLink *for* ADO data provider has the following format:

```
[DataDirect] [SequeLink ADO provider] message
```

For example:

```
[DataDirect] [SequeLink ADO provider] Invalid precision  
specified.
```

The native error code is always zero (0).

If you receive this type of error, check the last ADO call your application made. Contact your ADO or OLE DB application vendor, or refer to the ADO and OLE DB documentation available from Microsoft.

SequeLink® Client Errors

An error generated by the SequeLink *for* ADO Client has the following format:

```
[DataDirect] [SequeLink ADO provider] [SequeLink Client]  
message
```

For example:

```
[DataDirect] [SequeLink ADO provider] [SequeLink Client]  
Memory allocation error occurred.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

SequeLink® Server Errors

An error generated by SequeLink Server has the following format:

```
[DataDirect] [SequeLink ADO provider] [SequeLink Server]  
message
```

For example:

```
[DataDirect] [SequeLink ADO provider] [SequeLink Server]  
Only Select statements are allowed in this read-only  
connection.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

Database Errors

An error generated by the database has the following format:

```
[DataDirect] [SequeLink ADO provider] [...] message
```

For example:

```
[DataDirect] [SequeLink ADO provider] [Oracle]  
ORA-00942:table or view does not exist.
```

Use the native error code to look up details about the possible cause of the error. For these details, refer to your database documentation.

Part 3: Developing JDBC Applications

This part contains the following chapters:

- [Chapter 5 “Using the JDBC Client” on page 215](#) provides information about using JDBC applications with the SequeLink Client *for* JDBC.
- [Chapter 6 “Using DataDirect Test™” on page 263](#) introduces DataDirect Test, a tool that allows you to test and learn the JDBC API, and contains a tutorial that takes you through a working example of its use.
- [Chapter 7 “Tracking JDBC Calls” on page 315](#) introduces Spy, a tool that allows you to track JDBC calls, and describes how to use it.
- [Chapter 8 “Developing JDBC Applications” on page 327](#) provides information about developing JDBC applications for SequeLink environments.

5 Using the JDBC Client

This chapter provides information about using JDBC applications with the SequeLink Client *for* JDBC (the JDBC Client).

About the JDBC Client

The JDBC Client provides JDBC access through any Java-enabled applet, application, or application server. It delivers high-performance point-to-point and *n*-tier access to industry-leading data stores across the Internet and intranets. The JDBC Client is optimized for the Java environment, allowing you to incorporate Java technology and extend the functionality and performance of your existing system. The following components are shipped with the JDBC Client:

- SequeLink *for* JDBC Driver
- SequeLink Proxy Server
- DataDirect Spy™ *for* JDBC
- DataDirect Test™ *for* JDBC
- DataDirect Connection Pool Manager
- J2EE Connector Architecture resource adapters

The JDBC Client runs on 32-bit and 64-bit platforms. No changes are required to existing applications to enable them to run on 64-bit platforms.

JDBC Driver

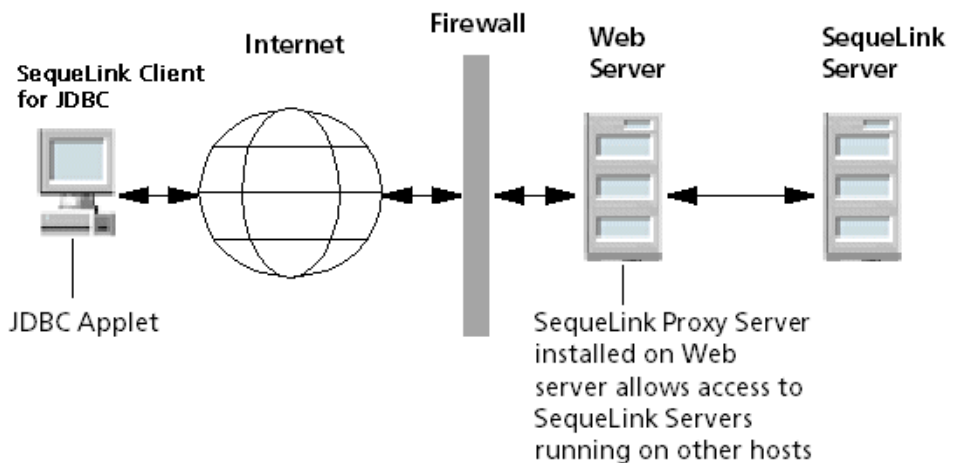
The JDBC driver is compliant with the JDBC 3.0 specification, including the following features:

- Java Naming Directory Interface (JNDI) for naming databases
- Connection pooling
- Distributed transactions

SequeLink® Proxy Server

Installing the SequeLink Proxy Server on the Web server from which your JDBC applets are downloaded allows untrusted applets to connect to SequeLink Servers on hosts other than the Web server, as shown in [Figure 5-1](#).

Figure 5-1. SequeLink Proxy Server Installed on a Web Server



In addition, you can use SSL encryption with the proxy server to encrypt data between the SequeLink Proxy Server and the JDBC Client. You can also use SSL with a Java application running on

your Intranet to secure data over your entire network by installing the SequeLink Proxy Server on the same machine as the SequeLink Server. For example, you may want to use SSL to encrypt the data sent between an application server and the data store serviced by a SequeLink Server on another machine. For more information about SSL, refer to the *SequeLink Administrator's Guide*.

DataDirect Spy™

DataDirect Spy is a software component for tracking JDBC calls at runtime. It passes calls issued by an application to an underlying JDBC driver and logs detailed information about those calls. DataDirect Spy provides the following advantages:

- Logging is JDBC 3.0-compliant, including support for the JDBC 2.0 Optional Package.
- Logging is consistent, regardless of the JDBC driver used.
- All parameters and function results for JDBC calls can be logged.
- Logging can be enabled without changing the application.
- DataDirect Spy can only be used with the SequeLink *for* JDBC Driver and the DataDirect Connect® *for* JDBC drivers.

When you enable DataDirect Spy for a connection, you can customize DataDirect Spy logging for your needs by setting one or multiple options for DataDirect Spy. For example, you may want to direct logging to a local file on your machine.

DataDirect Test™

DataDirect Test is a menu-driven software component that is included in the SequeLink package. It helps you debug your JDBC applications and learn how to use the JDBC driver. DataDirect Test contains menu selections that:

- Correspond to specific JDBC functions—for example, connecting to a database or passing a SQL statement.
- Encapsulate multiple JDBC function calls as a shortcut to perform some common tasks, such as displaying the contents of a result set.

DataDirect Test displays the results of all JDBC function calls in one window, while displaying fully commented, Java JDBC code in an alternate window.

DataDirect Connection Pool Manager

Database access performance can be improved significantly when connection pooling is used. *Connection pooling* means that connections are reused rather than created each time a connection is requested.

Your application can use connection pooling through the DataDirect Connection Pool Manager. See [Appendix D “JDBC Connection Pool Manager” on page 541](#) for more information about the DataDirect Connection Pool Manager.

J2EE Connector Architecture (JCA) Resource Adapter

The J2EE Connector architecture (JCA) defines a standard structure for connecting the J2EE platform to Enterprise Information Systems (EISs). JCA enables the integration of EISs with application servers and enterprise applications.

The JDBC Driver supports appropriate JDBC functionality through the JCA SPI by providing a resource adapter. The DataDirect resource adapter is provided in a resource archive (RAR) file, and is named like the JDBC driver files, for example, `sljc.rar`. See the Installed File list in the SequeLink readme file for the names and locations of the RAR files. See the *SequeLink Installation Guide* for information about creating the resource adapters. See [“J2EE Connector Architecture Resource Adapter” on page 232](#) for more information about using resource adapters with SequeLink.

JDBC Client Directory Structure

Table 5-1 shows the JDBC Client directory after installation and provides a description of the files.

Table 5-1. JDBC Client Directory and Files

Directories and Files	Description
books/*.*	Files for the SequeLink online books, which are in HTML format.
driver/examples/CheckAgainstCertificateFromFile.java driver/examples/CheckAgainstCertificateFromJar.java driver/examples/KeyStoreCertificateChecker.java	Contain Java source files that provide examples of certificate checkers.
driver/examples/JNDI_FILESYSTEM_Example.java driver/examples/JNDI_LDAP_Example.java	Contain Java source files that allow you to create JDBC data sources. These source files must be adapted for your environment, and subsequently compiled and run.
driver/lib/sljc.jar	JAR file containing all classes of the JDBC driver implementing the JDBC 3.0 API. To load the driver, add this path to your CLASSPATH variable. This JAR file also contains all classes of the JDBC driver implementing the JDBC 2.0 Optional Package. To use the JDBC 2.0 Optional Package, add this path to your CLASSPATH variable.
driver/lib/slssl14.jar driver/lib/iaik_jce_full.jar	JAR files required for J2SE 1.4.2 or higher JVMs. The files contain all classes of the JDBC driver that implement SSL encryption.
driver/lib/sljc.rar	Resource Archive for use with J2EE Connector Architecture.
help/*.*	Files for the HTML-based online help for the JDBC driver.

Table 5-1. JDBC Client Directory and Files (cont.)

Directories and Files	Description
testforjdbc/lib/testforjdbc.jar	JAR file containing all the DataDirect Test classes. To use DataDirect Test, add this path to your CLASSPATH variable.
testforjdbc/testforjdbc14.bat	Batch file that starts DataDirect Test.
testforjdbc/testforjdbc14.sh	UNIX shell script that starts DataDirect Test.
pool/lib/pool.jar	JAR file containing all the classes for the DataDirect Connection Pool Manager.
proxy/cmdsrv.exe	Executable that registers the SequeLink Proxy Server as a Windows service.
proxy/proxyserver14.bat	Batch file that starts the SequeLink Proxy Server.
proxy/cert/	Directory containing demo certificates.
proxy/demos/com/ddtek/sequelink/demo/demo.properties proxy/demos/com/ddtek/sequelink/demo/GenerateDemoCertificates\$DN.class proxy/demos/com/ddtek/sequelink/demo/GenerateDemoCertificates.class	Contain Java files you can use to generate certificates.
proxy/demos/com/ddtek/sequelink/demo/KeyTool.class	Contains a Java class file that extracts certificates from a Java2 KeyStore and converts certificates to different formats.
proxy/lib/slproxy.jar	JAR file containing all classes for the SequeLink Proxy Server.
proxy/log	The directory that contains all messages logged by the SequeLink Proxy Server.

Table 5-1. JDBC Client Directory and Files (cont.)

Directories and Files	Description
proxy/proxyserver14.sh	UNIX shell script that starts the SequeLink Proxy Server.
spy/lib/spy.jar	JAR file containing all Spy classes. To use DataDirect Spy, add this path to your CLASSPATH variable.
sun/lib/jdbc2_0-stdext.jar	JAR file containing redistributable Sun Microsystems components for the JDBC 2.0 Optional Package.
sun/lib/jndi.jar	JAR file containing redistributable Sun Microsystems components for JNDI 1.2.
sun/lib/jta-spec1_0_1.jar	JAR file containing redistributable Sun Microsystems components for JTA 1.0.1.
sun/lib/fs/fscontext.jar sun/lib/fs/providerutil.jar	JAR files containing redistributable Sun Microsystems components for the File System JNDI Provider.
sun/lib/ldap/jaas.jar sun/lib/ldap/ldap.jar sun/lib/ldap/ldapbp.jar sun/lib/ldap/providerutil.jar	JAR files containing redistributable Sun Microsystems components for the LDAP JNDI Provider.

Registering the JDBC Driver

To use the JDBC driver, you first must register it with the JDBC Driver Manager. You can register the JDBC driver in any of the following ways:

- **Method 1:** Set the Java property `jdbc.drivers` using the Java `-D` option. The `jdbc.drivers` property is defined as a colon-separated list of driver class names. For example:

```
com.ddtek.jdbc.sequellink.SequeLinkDriver:sun.jdbc.odbc.JdbcOdbcDriver
```

The `jdbc.drivers` property can be set like other Java properties, using the `-D` option. For example:

```
java -Djdbc.drivers=com.ddtek.jdbc.sequellink.SequeLinkDriver
```

- **Method 2:** Set the Java property `jdbc.drivers` from within your Java application or applet. To do this, include the following code in your application or applet, and call `DriverManager.getConnection()`:

```
Properties p = System.getProperties();  
p.put ("jdbc.drivers",  
"com.ddtek.jdbc.sequellink.SequeLinkDriver");  
System.setProperties (p);
```

- **Method 3:** Explicitly load the driver class using the standard `Class.forName()` method. To do this, include the following code in your application or applet and call `DriverManager.getConnection()`:

```
Class.forName("com.ddtek.jdbc.sequellink.SequeLinkDriver");
```

Specifying JDBC Driver Connection URLs

The connection URL format depends on whether you are using SSL encryption. For more information about SSL encryption, refer to the *SequeLink Administrator's Guide*.

NOTE: SequeLink Server for DB2 for z/OS does not support SSL. To use SSL encryption in a DB2 for z/OS environment, use the SequeLink Proxy Server.

If not using SSL encryption over the SequeLink Proxy Server, the connection URL format is:

```
jdbc:sequelink://hostname:port[;key=value]...
```

If using SSL encryption over the SequeLink Proxy Server, the connection URL format is:

```
jdbc:sequelink:ssl://hostname:port[;key=value]...
```

where:

<i>hostname</i>	is the TCP/IP address or TCP/IP host name of the SequeLink server to which you are connecting. NOTE: Untrusted applets cannot open a socket to a machine other than the originating host. For more information about untrusted applets, refer to the <i>SequeLink Administrator's Guide</i> .
<i>port</i>	is the TCP/IP port on which the SequeLink server is listening. A default installation of SequeLink Server uses the port 19996.
<i>key=value</i>	specifies connection properties. See "JDBC Connection Properties" on page 237 for a list of connection properties and their valid values.

JDBC Connection URL Examples:

The following examples show some typical JDBC driver connection URLs:

```
jdbc:sequelink://sequelinkhost:19996;

jdbc:sequelink://189.23.5.25:19996;user=john;
password=whatever

jdbc:sequelink://189.23.5.132:19996;databaseName=stores7

jdbc:sequelink://189.23.5.68:19996;databaseName=pubs;
HUser=john;HPassword=whatever

jdbc:sequelink://sequelinkhost:4006;
databaseName=pubs;DBUser=john;DBPassword=whatever

jdbc:sequelink:ssl://mysecurehost:9500;
cipherSuites=SSL_DH_anon_WITH_RC4_128_MD5

jdbc:sequelink:ssl://mysecurehost:9502;
cipherSuites=SSL_DHE_RSA_WITH_DES_CBC_SHA;
certificateChecker=CheckAgainstCertificateFromJar
```

The preceding examples do not show the user and password connection properties. Typically, these properties are specified in the connection properties stored in the `java.util.Properties` object, which is supplied as a parameter to the `getConnection` method.

Configuring JDBC Data Sources

Using JDBC data sources provides flexibility to make environment changes and reduces the time it takes to reconfigure your infrastructure when a change is made. For example, if a SequeLink service is reconfigured (for example, moved to another machine, port, and so on), the SequeLink administrator can change and run the configuration source file

described in [“Creating and Managing JDBC Data Sources” on page 226](#), reassigning the logical name of the JDBC data source to the changed data source configuration. As a result, the client application code does not have to change, because it only refers to the logical name of the JDBC data source.

SequeLink supports the following JDBC data source implementations defined by the JDBC 2.0 Optional Package:

- JNDI for Naming Databases
- Connection pooling
- Distributed Transaction Management Support

NOTES:

- You must include the `javax.sql.*` and `javax.naming.*` classes to create and use JDBC data sources. The JDBC Client provides all the necessary JAR files that contain the required classes and interfaces.
- In addition, you must include the `javax.transaction.xa.*` class to use and implement distributed transactions.

Creating and Managing JDBC Data Sources

JDBC data sources are implemented using a SequeLink class `com.ddtek.sequelink.jdbcx.datasource.SequeLinkDataSource`. This single data source implementation implements the following interfaces defined in the JDBC 2.0 Optional Package:

- `javax.sql.DataSource`
- `javax.sql.ConnectionPoolDataSource`
- `javax.sql.XADataSource`

The SequeLink Data Source implementation implements both the `java.io.Serializable` and `javax.naming.Referenceable` interfaces. The interface that is used depends on the service provider you are

using and how the `SequeLinkDataSource` object is saved in your JNDI environment.

Your JDBC Client installation contains the following examples that show how to create and use JDBC data sources:

- `JNDI_LDAP_Example.java`. Use this example to create a JDBC data source and save it in your LDAP directory, using the JNDI Provider for LDAP.
- `JNDI_FILESYSTEM_Example.java`. Use this example to create a JDBC data source and save it in your local file system, using the File System JNDI Provider.

NOTE: You must include the `javax.sql.*` and `javax.naming.*` classes to create and use `SequeLink` JDBC data sources. The `SequeLink` JDBC driver provides all the necessary JAR files, which contain the required classes and interfaces.

Calling a Data Source in an Application

Applications can call a `SequeLink` JDBC data source using a logical name to retrieve the `javax.sql.DataSource` object. This object loads the specified driver and can be used to establish a connection to the database.

Once the data source has been registered with JNDI, it can be used by your JDBC application as shown in the following example:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("EmployeeDB");
Connection con = ds.getConnection("scott", "tiger");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the data source (`EmployeeDB`). The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.DataSource` object. Finally, the

`DataSource.getConnection()` method is called to establish a connection with the database.

See [“Creating and Managing JDBC Data Sources” on page 226](#) for information about example data sources shipped with SequeLink Client for JDBC that you can use as a template for creating your own data sources.

Using JNDI for Naming Databases

Instead of using connection URLs, client applications can access a JNDI-named data source using a logical name to retrieve the `javax.sql.DataSource` object. This object loads the JDBC driver and establishes the connection to the SequeLink service.

Once a JDBC data source has been registered with JNDI, it can be used by your JDBC application as shown in the following example:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/EmployeeDB");
Connection con = ds.getConnection("scott", "tiger");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the JDBC data source. The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.DataSource` object. Finally, the `DataSource.getConnection()` method is called to establish a connection with the SequeLink service.

See [“Creating and Managing JDBC Data Sources” on page 226](#) for instructions on creating JDBC data sources.

Using Connection Pooling

Connection pooling allows you to reuse connections rather than create a new one every time the SequeLink Client needs to establish a data access connection. Connection pooling manages connection sharing across different user requests to maintain performance and reduce the number of new connections that must be created. For example, compare the transaction sequences shown in [“Example A: Without Connection Pooling” on page 229](#) and [“Example B: With Connection Pooling” on page 229](#).

Example A: Without Connection Pooling

- 1 The client application creates a connection.
- 2 The client application sends a data access query.
- 3 The client application obtains the result set of the query.
- 4 The client application displays the result set to the end user.
- 5 The client application ends the connection.

Example B: With Connection Pooling

- 1 The client checks the connection pool for an unused connection.
- 2 If an unused connection exists, it is returned by the pool implementation; otherwise, it creates a new connection.
- 3 The client application sends a data access query.
- 4 The client application obtains the result set of the query.
- 5 The client application displays the result set to the end user.
- 6 The client application returns the connection to the pool.

NOTE: The client application still calls `close()`, but the connection remains open and the pool is notified of the close request.

The pool implementation creates real database connections using the `getPooledConnection()` method of `ConnectionPoolDataSource`. Then, the pool implementation registers itself as a listener to the `PooledConnection`. When a client application requests a connection, the pool implementation is notified by the `ConnectionEventListener` interface that the connection is free and available for reuse. The pool implementation is also notified by the `ConnectionEventListener` interface when the client somehow corrupts the database connection, so that the pool implementation can remove that connection from the pool.

Once a JDBC data source has been registered with JNDI, it can be used by your JDBC application as shown in the following example, typically through a third-party connection pool tool:

```
Context ctx = new InitialContext();
ConnectionPoolDataSource ds = (ConnectionPoolDataSource)
ctx.lookup("jdbc/EmployeeDB");
pooledConnection pcon = ds.getPooledConnection("scott",
"tiger");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the JDBC data source. The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.ConnectionPoolDataSource` object. Finally, the `ConnectionPoolDataSource.getPooledConnection()` method is called to establish a connection with the SequeLink service.

See [“Creating and Managing JDBC Data Sources” on page 226](#) for instructions on creating JDBC data sources. See [Appendix D “JDBC Connection Pool Manager” on page 541](#) for more information on the DataDirect Connection Pool Manager.

Using the Java Transaction API

[Table 5-2](#) lists which databases are supported for the Java Transaction API (JTA) by the JDBC Client.

Table 5-2. Support for the Java Transaction API (JTA) by the JDBC Client

Database	JTA Supported?
Oracle	Yes
Informix	Yes
DB2 UDB on z/OS	Yes
DB2 UDB on Linux, UNIX, and Windows	Yes
JDBC Socket	No
Microsoft SQL Server	Yes
ODBC Socket	No
Sybase	Yes

Once a JDBC data source has been registered with JNDI, it can be used by your JDBC application as shown in the following example, typically through application server software:

```
Context ctx = new InitialContext();
XADataSource ds = (XADataSource)
ctx.lookup("jdbc/EmployeeDB");
XAConnection xacon = ds.getXAConnection("scott", "tiger");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the JDBC data source. The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.XADataSource` object. Finally, the

`XADataSource.getXAConnection()` method is called to establish a connection with the SequeLink service.

See [“Creating and Managing JDBC Data Sources” on page 226](#) for instructions on creating JDBC data sources.

J2EE Connector Architecture Resource Adapter

The J2EE Connector Architecture defines a standard structure for connecting the J2EE platform to Enterprise Information Systems (EIS). Examples of EIS include mainframe transaction processing, database systems, and legacy applications that are not written in the Java programming language. The J2EE Connector Architecture allows you to integrate EIS with application servers and enterprise applications. The J2EE Connector Architecture defines a standard set of system-level contracts between an application server and the EIS to ensure compatibility between them. The resource adapter implements the EIS portion of these system-level contracts.

The J2EE Connector Architecture also defines a standard Service Provider Interface (SPI) for integrating the transaction, security and connection management facilities of an application server with those of a transactional resource manager. The JDBC specification provides more information about the relationship of JDBC to the SPI specified in the J2EE Connector Architecture.

The JDBC driver supports JDBC functionality through the J2EE Connector Architecture SPI by providing resource adapters. A *resource adapter* is a system-level software driver used by an application server to connect to an EIS. The resource adapter communicates with the server to provide the underlying transaction, security, and connection pooling mechanisms.

The SequeLink resource adapter conforms to the J2EE Connector Architecture 1.0 specification. The resource adapter is provided in

a resource archive (RAR) file, `sljc.rar`. Refer to the *SequeLink Installation Guide* for information about creating the resource adapter.

Using the Resource Adapter with an Application Server

The SequeLink resource adapter can be used with any J2EE 1.3 or higher application server. To use the resource adapter with J2EE 1.4 or higher application servers, it must be used in conjunction with the Sun Microsystems JDBC Connector. Refer to the Sun Microsystems JDBC Connector documentation for more information or see the following Web site:

<http://java.sun.com/developer/earlyAccess/jdbc/index.html>

In an application server environment, the resource adapter is deployed using a deployment tool. Each RAR file includes a *deployment descriptor*, which instructs the application server about how to use the resource adapter in an application server environment. The deployment descriptor contains information about the resource adapter, including security and transactional capabilities, and the `ManagedConnectionFactory` class name. Refer to your application server documentation for details about how to deploy components using the deployment tool.

Using the Resource Adapter from an Application

The JCA resource adapter may also be used directly from an application, rather than through a container-managed, application server environment. The following code example shows how you might access a database using the resource adapter:

```

package examples;

import java.util.Hashtable;
import java.sql.Connection;
import javax.sql.DataSource;
import javax.naming.*;
import javax.resource.spi.*;
import com.ddtek.resource.sljdbc.JCAConnectionFactory;
import com.ddtek.resource.sljdbc.spi.*;

public class RAExample {
    static public void main(String[] args) {
        try {
            // Create a connection factory instance
            SequeLinkManagedConnectionFactory managedFactory =
                new SequeLinkManagedConnectionFactory();
            managedFactory.setServerName("MyOracleServer");
            managedFactory.setPortNumber("1521");
            JCAConnectionFactory factory = (JCAConnectionFactory)

managedFactory.createConnectionFactory();
            // Get an InitialContext. Using File System JNDI Service
            // Provider as an example
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.fscontext.RefFSContextFactory");
            env.put(Context.PROVIDER_URL,
                "file:c:/ConnectionFactoryies");
            Context connectorContext = new InitialContext(env);
            // Bind the connection factory
            try {
                connectorContext.bind("ConnectionFactory",
factory);
            } catch (NameAlreadyBoundException except) {
                connectorContext.rebind("ConnectionFactory",
                    factory);
            }
        } catch (Exception except) {
            System.out.println("Error creating DataSource");
            System.exit(0);
        }
    }
}

```

```
// Connect via the DataSource
    try {
// Get an InitialContext. Using File System JNDI Service
// Provider as an example
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,

"com.sun.jndi.fscontext.RefFSContextFactory");
        env.put(Context.PROVIDER_URL,
            "file:c:/ConnectionFactory");
        Context connectorContext = new InitialContext(env);
// Lookup the connection factory
        DataSource dataSource = (DataSource)

connectorContext.lookup("ConnectionFactory");
Connection connection =dataSource.getConnection("scott",
"tiger");
        catch (Exception except) {
            System.out.println("Error looking up connection
factory"); } }
    }
```

Specifying Connection Properties

You can specify connection properties using a connection URL, the JDBC Driver Manager, or JDBC data sources. The properties you can specify depend on the connection method you choose. See [“JDBC Connection Properties” on page 237](#) for a list of the connection properties.

Using Connection URLs or the JDBC Driver Manager

In order of precedence, you can specify connection properties using:

- `getConnection(url, user, password)`, where *user* and *password* are specified using the `getConnection` method defined in `java.sql.DriverManager`.
- `java.util.properties` object.
- Connection URL specified using the URL parameter of the `getConnection` method defined in `java.sql.DriverManager`.
- Server data sources specified using the SequeLink Manager.

For more information about server data sources, refer to the *SequeLink Administrator's Guide*.

Using JDBC Data Sources

In order of precedence, you can specify connection properties using:

- `getConnection(user, password)`, where *user* and *password* are specified using the `getConnection` method defined in `javax.sql.DataSource`

```
getConnection("scott", "tiger")
```

- JDBC `DataSource` object
- Server data sources specified using the SequeLink Manager

For more information about server data sources, refer to the *SequeLink Administrator's Guide*.

JDBC Connection Properties

[Table 5-3](#) lists the JDBC connection properties supported by the JDBC driver, describes each property, and specifies the methods with which it can be specified.

Table 5-3. JDBC Connection Properties

Property	Description
allowPrefetch	<p>allowPrefetch={0 1}. Enables the prefetch feature. When enabled, the JDBC driver requests a next set of rows from the server while the client application is processing the previous set of rows.</p> <p>When set to 1, the prefetch feature is enabled. Overall throughput increases, if the application always fetches all rows from a result set. When this feature is enabled and the application does not fetch all data from result sets, performance can be significantly degraded.</p> <p>When set to 0 (the initial default), the prefetch feature is disabled.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
AlternateServers	<p>A comma-separated list of alternate SequeLink servers that the driver will try to connect to if the primary SequeLink server is unavailable. The value of this property is a string that specifies each alternate SequeLink server. This string has the format:</p> <pre>(servername1:port1[;serverDataSource=datasourcename], servername2:port2[;serverDataSource=datasourcename],...)</pre> <p>The server name and port number are required for each alternate server entry. The serverDataSource property (serverDataSource=datasourcename) is optional for each alternate SequeLink server entry.</p> <p>For example, the connection string</p> <pre>jdbc:sequelink://server1:19996;serverDataSource=SDSN1;User=test;Password=secret;AlternateServers=(server2:19996;serverDataSource=SDSN2,server3:19996;serverDataSource=SDSN3)</pre> <p>contains alternate server entries for server2 and server3.</p> <p>The ConnectionRetryCount property controls the number of times the driver retries the primary database server, and if specified, alternate servers while attempting to establish a connection. The ConnectionRetryDelay property sets the wait interval, in seconds, between retry attempts.</p> <p>The LoadBalancing property controls the order in which the driver sequences through the list of servers (primary and alternate) while attempting to establish a connection.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
ApplicationName	<p>Identifies the application that is establishing the connections. When the application does not provide a value, the initial default is SequeLink for JDBC Application.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties
blockFetchForUpdate	<p>blockFetchForUpdate={0 1}. Specifies a workaround connection attribute. When the isolation level is Read Committed and a SELECT FOR UPDATE statement is issued against some data stores, the JDBC Client does not lock the expected row.</p> <p>When set to 0, the appropriate row is locked.</p> <p>When set to 1 (the initial default), the appropriate row is not locked.</p> <p>IMPORTANT: Specifying 0 will degrade performance for SELECT FOR UPDATE statements because rows will be fetched one at a time.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
certificateChecker	<p>The fully qualified class name of a user-defined server certificate checker class. When the SequeLink Client and SequeLink Server have agreed on an SSL cipher suite that requires a server certificate, this class is used to verify the server certificate on behalf of the client. The class must be an implementation of the <code>com.ddtek.sequelink.cert.CertificateCheckerInterface</code> interface.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ <code>java.util.properties</code> <p>For more information about certificate checker classes, refer to the <i>SequeLink Administrator's Guide</i>.</p>
cipherSuites	<p>The SSL cipher suites with which the JDBC Client can use to connect. This property is required when <code>networkProtocol=ssl</code>.</p> <p>For a list of supported cipher suites, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ <code>java.util.properties</code>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
ConnectionRetryCount	<p>The number of times the driver retries connection attempts to the primary SequeLink server, and if specified, alternate SequeLink servers until a successful connection is established. Valid values are 0 and any positive integer.</p> <p>When set to 0 (the initial default), the driver does not try to reconnect after the initial unsuccessful attempt.</p> <p>For example, in the case where the following properties are specified:</p> <pre>AlternateServers=(server2:19996,server3:19996,server4:19999);</pre> <p>and</p> <pre>ConnectionRetryCount=1</pre> <p>If a connection is not successfully established on the driver's first pass through the list of database servers, the driver retries all the servers in the list only once.</p> <p>If an application sets a login timeout value (for example, using <code>DataSource.loginTimeout</code> or <code>DriverManager.loginTimeout</code>), the login timeout takes precedence over this property. For example, if the login timeout expires, any connection attempts stop.</p> <p>The <code>ConnectionRetryDelay</code> property sets the wait interval, in seconds, between retry attempts.</p> <p>If the <code>LoadBalancing</code> property is set to true, the driver sequence through the list of servers (primary and alternate) in a different order each time.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
ConnectionRetryDelay	<p>The number of seconds the driver will wait between connection retry attempts when ConnectionRetryCount is set to a positive integer.</p> <p>The initial default is 3.</p> <p>For example, in the case where the following properties are specified:</p> <pre>AlternateServers= (server2:19996,server3:19996,server4:19996)</pre> <p>and</p> <pre>ConnectionRetryCount=2</pre> <p>and</p> <pre>ConnectionRetryDelay=5</pre> <p>If a connection is not successfully established on the driver's first pass through the list of SequeLink servers, the driver retries the list of servers twice. It waits 5 seconds between the first connection retry attempt and the second connection retry attempt.</p>
ConvertNull	<p>ConvertNull={1 0}. Controls how data conversions are handled for null values.</p> <p>If set to 1 (the default), the driver checks the data type being requested against the data type of the table column storing the data. If a conversion between the requested type and column type is not defined, the driver generates an "unsupported data conversion" exception regardless of the data type of the column value.</p> <p>If set to 0, the driver does not perform the data type check if the value of the column is null. This allows null values to be returned even though a conversion between the requested type and the column type is undefined.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
databaseName	<p>The name of the data store to which you want to connect.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
DBPassword	<p>The data store password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties
DBUser	<p>The data store user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties
description	<p>A description of the connection or data source.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
encrypted	<p>encrypted={0 1}. Enables the use of direct SSL encryption. When SSL encryption is configured on the SequeLink Server, only this setting is required. When SSL encryption is not configured on the SequeLink Server, this property is ignored.</p> <p>NOTE: Because SequeLink Server for DB2 for z/OS cannot support SSL, enabling this property in the connection string generates an error.</p> <p>When set to 0 (the default), direct SSL encryption is not used.</p> <p>When set to 1, direct SSL encryption is used.</p> <p>Example:</p> <pre>jdbc:sequelink://mysecurehost:19996;encrypted=1</pre>
HPassword	<p>The host password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ JDBC data source■ URL■ java.util.properties
HUser	<p>The host user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ JDBC data source■ URL■ java.util.properties

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
InitializationString	<p>Specifies one or multiple SQL commands to be executed by the driver after it has established the connection to the database and has performed all initialization for the connection. The following connection URL sets the handling of null values to the SequeLink default:</p> <pre>jdbc:datadirect:sybase://server1:5000; InitializationString=set ANSINULL off; DatabaseName=test</pre> <p>Multiple commands must be separated by semicolons. In addition, if this property is specified in a connection URL, the entire value must be enclosed in parentheses when multiple commands are specified. The following connection URL sets the handling of null values to the Sybase default and allows delimited identifiers:</p> <pre>jdbc:datadirect:sybase://server1:5000; InitializationString=(set ANSINULL off; set QUOTED_IDENTIFIER on);DatabaseName=test</pre> <p>If the execution of a SQL command fails, the connection attempt also fails and the driver throws an exception indicating which SQL command or commands failed.</p>

Table 5-3. JDBC Connection Properties *(cont.)*

Property	Description
<code>insensitiveResultSetBufferSize</code>	<p><code>insensitiveResultSetBufferSize={-1 0 x}</code>. Determines the amount of memory used by the driver to cache insensitive result set data. It must have one of the following values:</p> <p>If set to -1, the driver caches all insensitive result set data in memory. If the size of the result set exceeds available memory, an <code>OutOfMemoryException</code> is generated. Because the need to write result set data to disk is eliminated, the driver processes the data more efficiently.</p> <p>If set to 0, the driver caches all insensitive result set data in memory, up to a maximum of 2 GB. If the size of the result set data exceeds available memory, the driver pages the result set data to disk. Because result set data may be written to disk, the driver may have to reformat the data to write it correctly to disk.</p> <p>If set to x, where x is a positive integer, the driver caches all insensitive result set data in memory, using this value to set the size (in KB) of the memory buffer for caching insensitive result set data. If the size of the result set data exceeds the buffer size, the driver pages the result set data to disk. Because the result set data may be written to disk, the driver may have to reformat the data to write it correctly to disk. Specifying a buffer size that is a power of 2 results in more efficient memory use.</p> <p>The initial default is 2048 (KB).</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
JavaDoubleToString	<p>JavaDoubleToString={true false}. Determines whether the driver uses its internal conversion algorithm or the JVM conversion algorithm when converting double or float values to string values.</p> <p>If set to true, the driver uses the JVM algorithm when converting double or float values to string values.</p> <p>If set to false (the initial default), the driver uses its internal algorithm when converting double or float values to string values. Using this value improves performance; however, slight rounding differences can occur when compared to the same conversion using the JVM algorithm. These differences are within the allowable error of the double and float data types.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
LoadBalancing	<p>LoadBalancing={true false}. Determines whether the driver will use client load balancing in its attempts to connect to the list of SequeLink servers (primary and alternate).</p> <p>When set to true, client load balancing is used and the driver attempts to connect to the list of SequeLink servers (primary and alternate servers) in random order.</p> <p>When set to false (the initial default), client load balancing is not used and the driver connects to each server based on their sequential order (primary server first, then, alternate servers in the order they are specified).</p> <p>For example, in the case where the following properties are specified:</p> <pre>AlternateServers= (server2:19996,server3:19996,server4:19996)</pre> <p>and</p> <pre>LoadBalancing=true,</pre> <p>The driver randomly selects from the list of primary and alternate servers which server to connect to first. If that connection fails, the driver again randomly selects from this list of servers until all servers in the list have been tried or a connection is successfully established.</p> <p>Refer to the <i>SequeLink Administrator's Guide</i> for a discussion of specifying connection information for primary and secondary SequeLink servers.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
LongDataCacheSize	<p>LongDataCacheSize={-1 0 x}. Determines whether the driver caches long data in result sets (images, pictures, long text, or binary data).</p> <p>If set to -1, the driver does not cache long data in result sets. It is cached on the server. Use this value only if your application returns columns in the order they are defined in the result set.</p> <p>If set to 0, the driver caches long data in result sets in memory. If the size of the result set data exceeds available memory, the driver pages the result set data to disk.</p> <p>If set to x, where x is a positive integer, the driver caches long data in result sets in memory and uses this value to set the size (in KB) of the memory buffer for caching result set data. If the size of the result set data exceeds available memory, the driver pages the result set data to disk.</p>
MSSMapLongtoDecimal	<p>Turns on client-side workarounds that allow you to take full advantage of the JDBC driver with JDBC applications that require non-standard or extended behavior. For more information, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Connection Properties *(cont.)*

Property	Description
networkProtocol	<p>networkProtocol={socket ssl}. Specifies the protocol to be used.</p> <p>The initial default is socket.</p> <p>When set to socket (the initial default), SSL encryption is not used.</p> <p>When set to ssl, SSL encryption over the SequeLink Proxy Server is used. This has the same effect as specifying the following:</p> <pre>jdbc:sequelink:ssl://host= ...</pre> <p>NOTE: This property is not required when using the encrypted connection property to define direct SSL encryption without the use of the SequeLink Proxy Server.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ JDBC data source■ URL■ java.util.properties
newPassword	<p>The new host password to be used. If specified and applicable to the connection, the SequeLink password change mechanism is invoked. When the password has been changed successfully, the following warning is returned:</p> <pre>[DataDirect][SequeLink JDBC driver] [SequeLink Server] The user password was changed successfully</pre> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ JDBC data source■ URL■ java.util.properties <p>For more information about the SequeLink password change mechanism, refer to the <i>SequeLink Administrator's Guide</i>.</p>

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
ORANumberOfIsNumeric	<p>Turns on client-side workarounds that allow you to take full advantage of the JDBC driver with JDBC applications that require non-standard or extended behavior. For more information, refer to the <i>SequeLink Administrator's Guide</i>.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
password	<p>The host or data store password, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ getConnection ■ JDBC data source ■ URL ■ java.util.properties
portNumber	<p>The TCP/IP port on which the SequeLink service is listening.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL
QueryTimeout	<p>QueryTimeout={0 -1}. The value of this option will set the default query timeout (in seconds) for all statements created by the connection.</p> <p>When set to 0 (the initial default), there is no timeout. If an application calls the Statement.setQueryTimeout method to set a timeout value for a statement, the value specified in the setQueryTimeout method will override the default value specified by this connection option.</p> <p>When set to -1, the driver query timeout functionality is disabled. The driver silently ignores calls to Statement.setQueryTimeout.</p>

Table 5-3. JDBC Connection Properties *(cont.)*

Property	Description
resultSetMetaDataOptions	<p>resultSetMetaDataOptions={0 1}. Returns table name information in the ResultSet metadata for Select statements if your application requires that information.</p> <p>If set to 0 (the initial default) and the ResultSetMetaData.getTableName() method is called, the driver does not perform additional processing to determine the correct table name for each column in the result set. In this case, the getTableName() method may return an empty string for each column in the result set.</p> <p>If set to 1 and the ResultSetMetaData.getTableName() method is called, the driver performs additional processing to determine the correct table name for each column in the result set. The driver also can return schema name and catalog name information when the ResultSetMetaData.getSchemaName() and ResultSetMetaData.getCatalogName() methods are called if the driver can determine that information.</p> <p>By default, the JDBC driver skips the additional processing required to return the correct table name for each column in the result set when the ResultSetMetaData.getTableName() method is called. Because of this, the getTableName() method may return an empty string for each column in the result set. If you know that your application does not require table name information, this setting provides the best performance.</p>
serverDataSource	<p>A property that specifies a string to identify the server data source to be used for the connection. If unspecified, the configuration of the default server data source will be used for the connection.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ JDBC data source■ URL■ java.util.properties

Table 5-3. JDBC Connection Properties (cont.)

Property	Description
serverName	<p>The TCP/IP address of the SequeLink server in dotted format or host name format.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL
SLKStaticCursorLongColBuffLen	<p>The amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with an insensitive result set.</p> <p>The initial default is 4.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none"> ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
spyAttributes	<p>Enables DataDirect Spy, a tool that can be used to log detailed information about calls issued by a running application to the SequeLink for JDBC driver. The format for the value of this property is:</p> <pre>(spy_attribute[;spy_attribute]...)</pre> <p>where spy_attribute is any valid DataDirect Spy attribute. See Chapter 7 "Tracking JDBC Calls" on page 315 for a list of supported attributes.</p> <p>For example:</p> <pre>SpyAttributes=(log=(file)/tmp/spy.log;linelimit=80)</pre> <p>logs all JDBC activity to a file using a maximum of 80 characters for each line.</p> <p>NOTE: If coding a path on Windows to the log file in a Java string, the backslash character (\) must be preceded by the Java escape character, a backslash. For example:</p> <pre>log=(file)C:\\temp\\spy.log</pre> <p>By default, DataDirect Spy is not enabled.</p>

Table 5-3. *JDBC Connection Properties* (cont.)

Property	Description
transliterationWorkaroundServer	<p>transliterationWorkaroundServer={0 1 2}. Turns on a workaround for multiple transliteration workarounds. Refer to the <i>SequeLink Administrator's Guide</i> for more information about how SequeLink handles transliteration.</p> <p>When set to 0 (the initial default), the workaround is not enabled.</p> <p>When set to 1 or 2, this workaround resolves transliteration issues between Shift-JIS/Windows-31j and eucJP by mapping "look-alike" characters.</p>
user	<p>The host or data store user name, which may be required depending on the server configuration.</p> <p>This property can be specified using:</p> <ul style="list-style-type: none">■ getConnection■ JDBC data source■ URL■ java.util.properties

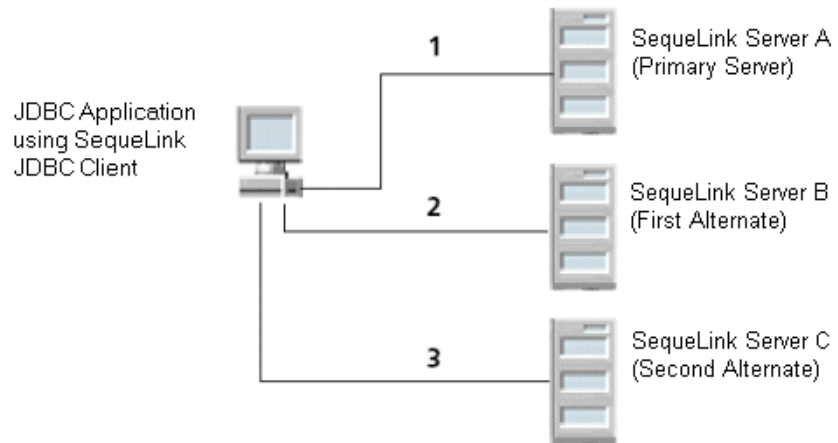
Configuring Connection Failover

Connection failover allows an application to connect to an alternate, or backup, SequeLink Server if the primary SequeLink Server is unavailable, for example, because of a hardware failure or traffic overload. Connection failover ensures that the data on which your critical JDBC applications depend is always available.

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

For example, suppose you have the environment shown in [Figure 5-2](#) with multiple SequeLink servers: SequeLink server A, B, and C. SequeLink server A is designated as the primary SequeLink server, SequeLink server B is the first alternate server, and SequeLink Server C is the second alternate server.

Figure 5-2. Connection Failover Using the JDBC Client



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

In this scenario, it is probable that at least one connection attempt would succeed, but if no connection attempt succeeds, the SequeLink Client can retry each alternate SequeLink server (primary and alternate) for a specified number of attempts.

To configure connection failover, you **must** specify a list of alternate SequeLink servers that are tried at connection time if

the primary server is not accepting connections. To do this, use the `AlternateServers` connection property. Connection attempts continue until a connection is successfully established or until all the SequeLink servers in the list have been tried once (the default).

Optionally, you can specify the following additional connection failover features:

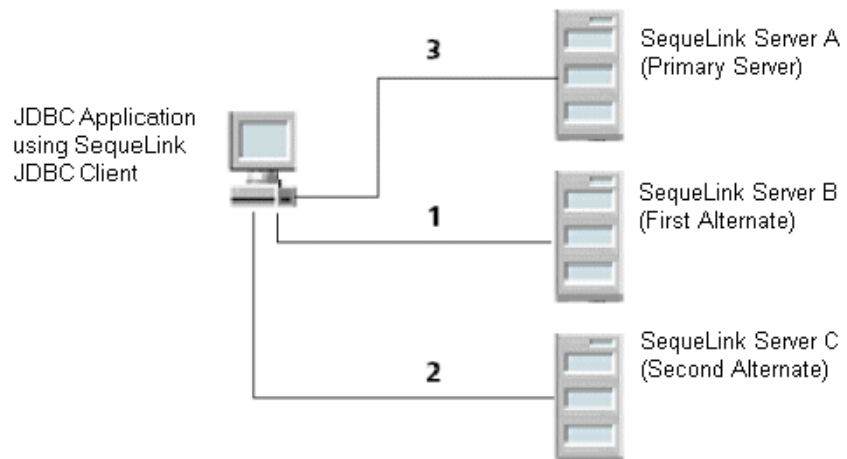
- The number of times the JDBC Client attempts to connect to the primary and alternate SequeLink servers after the initial unsuccessful connection attempt. By default, the JDBC Client does not retry. To set this feature, use the `ConnectionRetryCount` connection property. See [“Using Connection Retry” on page 258](#) for more information.
- The wait interval, in seconds, between attempts to connect to the primary and alternate SequeLink servers. The default interval is 3 seconds. To set this feature, use the `ConnectionRetryDelay` connection property.
- Whether the JDBC Client will use client load balancing in its attempts to connect to primary and alternate SequeLink servers. If load balancing is enabled, the JDBC Client uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load balancing. To set this feature, use the `LoadBalancing` connection property. See [“Using Client Load Balancing” on page 257](#) for more information.

You can use a connection URL to direct the JDBC to use connection failover, or use a JNDI LDAP provider. Refer to the *SequeLink Administrator's Guide* for detailed information.

Using Client Load Balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When client load balancing is enabled, the order in which primary and alternate SequeLink servers are tried is random. For example, let us suppose that client load balancing is enabled as shown in [Table 5-3](#):

Figure 5-3. Client Load Balancing Using the JDBC Driver



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

For details on configuring client load balancing, refer to the *SequeLink Administrator's Guide*.

Using Connection Retry

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

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

Using connection options, you can specify the number of times the driver attempts to connect and the time in seconds between connection attempts. For details on configuring connection retry, see ["Using Connection Retry" on page 258](#).

Testing JDBC Connections

See [Chapter 6 "Using DataDirect Test™" on page 263](#) for instructions on connecting with the JDBC Client using DataDirect Test.

Using the JDBC Client on a Java 2 Platform

When using the JDBC driver on a Java 2 Platform with the standard security manager enabled, you must give the driver some additional permissions. Refer to your Java 2 Platform documentation for more information about the Java 2 Platform security model and permissions.

You can run an application on a Java 2 Platform with the standard security manager using:

```
"java -Djava.security.manager application_class_name"
```

where *application_class_name* is the class name of the application.

Web browser applets running in the Java 2 plug-in are always running in a JVM with the standard security manager enabled. To enable the necessary permission, you must add them to the security policy file of the Java 2 Platform. This security policy file can be found in the `jre\lib\security` subdirectory of the Java 2 Platform installation directory.

To use JDBC data sources, all code bases must have the following permissions:

```
// permissions granted to all domains
grant {
    // DataSource access
    permission java.util.PropertyPermission "java.naming.*",
        "read,write";
    // Adjust the server host specification for your environment
    permission java.net.SocketPermission "*.ddtek.be:0-65535",
        "connect";
};
```

To use scroll-insensitive scrollable cursors, all code bases must have access to temporary files:

```
// permissions granted to all domains
grant {
// Permission to create and delete temporary files.
// Adjust the temporary directory for your environment.
permission java.io.FilePermission "C:\\TEMP\\-",
"read,write,delete";
};
```

To use SSL or other data privacy functionality, the following permissions are required for the JDBC Client code base only:

```
// permissions granted to the SequeLink JDBC Client code base only
grant codeBase "file:/slje/lib/-" {
// Security providers
// Only needed when using SSL or other data privacy functionality
// (e.g. fixed key DES/3DES)
permission java.security.SecurityPermission
"putProviderProperty.IAIK";
permission java.security.SecurityPermission "insertProvider.IAIK";
};
```

Applets that connect to another server other than the one they are downloaded from must have the following permission:

```
// permissions granted to the SequeLink JDBC Client code base only
grant codeBase "file:/slje/lib/-" {
// TCP/IP

// Adjust the server host specification for your environment
permission java.net.SocketPermission "*.ddtek.be:0-65535",
"connect";
};
```

NOTES:

- Make sure that you adjust the code base of the JDBC Client for your environment. For an applet, this will probably start with `http://` or `https://`.
- Make sure you adjust the server host specification and location of temporary files for your environment.

6 Using DataDirect Test™

This chapter provides information about DataDirect Test, a tool that allows you to test and learn the JDBC API, and contains a tutorial that takes you through a working example of its use.

DataDirect Test contains menu selections that correspond to specific JDBC functions—for example, connecting to a database or passing a SQL statement. It allows you to:

- Execute a single JDBC method or execute multiple JDBC methods simultaneously, so that you can easily perform some common tasks, such as returning result sets
- Display the results of all JDBC function calls in one window, while displaying fully commented, Java JDBC code in an alternate window

DataDirect Test™ Tutorial

This DataDirect Test tutorial explains how to use the most important features of DataDirect Test (and the JDBC API) and assumes that you can connect to a database with the standard available demo table or fine-tune the sample SQL statements shown in this example as appropriate for your environment.

NOTE: The step-by-step examples used in this tutorial do not show typical clean-up routines (for example, closing result sets and connections). These steps have been omitted to simplify the examples. Do not forget to add these steps when you use equivalent code in your applications.

Configuring DataDirect Test™

The default DataDirect Test configuration file is:

install_dir/testforjdbc/Config.txt

where *install_dir* is your SequeLink *for* JDBC Driver installation directory. This file can be edited as appropriate for your environment using any text editor. All parameters are configurable, but the most commonly configured parameters are:

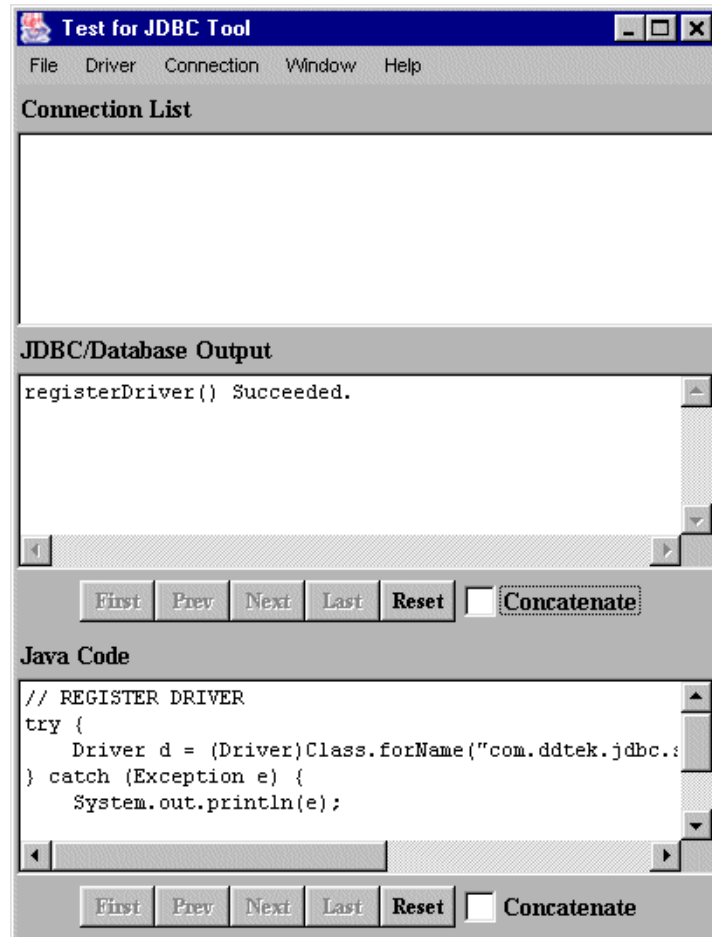
Drivers	A list of colon-separated JDBC driver classes.
DefaultDriver	The default JDBC driver that appears in the Get Driver URL window.
Databases	A list of comma-separated JDBC URLs. The first item in the list appears as the default in the database selection window. You can use one of these URLs as a template when you make a JDBC connection. The default Config.txt file contains example URLs for most databases.
InitialContextFactory	Should be set to com.sun.jndi.fscontext.RefFSContextFactory if you are using file system data sources, or com.sun.jndi.ldap.LdapCtxFactory if you are using LDAP.
ContextProviderURL	The location of the .bindings file if you are using file system data sources, or your LDAP Provider URL if you are using LDAP.
Datasources	A list of comma-separated JDBC data sources. The first item in the list appears as the default in the data source selection window.

Starting DataDirect Test™

How you start DataDirect Test depends on your platform:

- **As a Java application on Windows**—Run the testforjdbc.bat file located in the testforjdbc directory.
- **As a Java application on UNIX**—Run the testforjdbc.sh shell script located in the testforjdbc directory beneath the installation directory.

After you start DataDirect Test, the following window appears:



The main DataDirect Test window shows the following information:

- In the Connection List box, a list of available connections.
- In the JDBC/Database scroll box, a report indicating whether the last action succeeded or failed.
- In the Java Code scroll box, the actual Java code used to implement the last action.

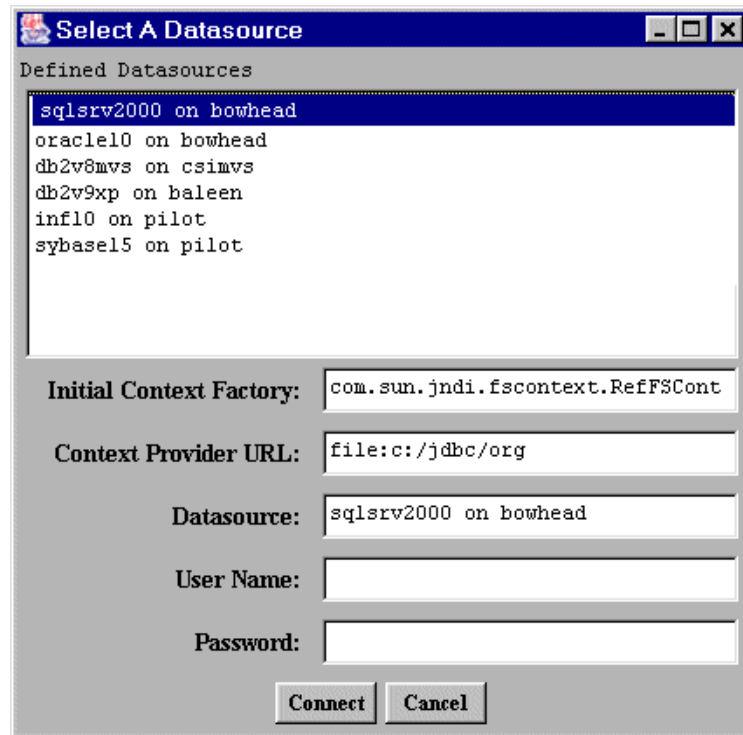
TIP: The DataDirect Test windows contain two Concatenate check boxes. Select a Concatenate check box to see a cumulative record of previous actions; otherwise, only the last action is shown. Selecting Concatenate can degrade performance, particularly when displaying large resultSets.

Connecting Using DataDirect Test™

There are two methods to connect using DataDirect Test: through a data source or through driver/database selection.

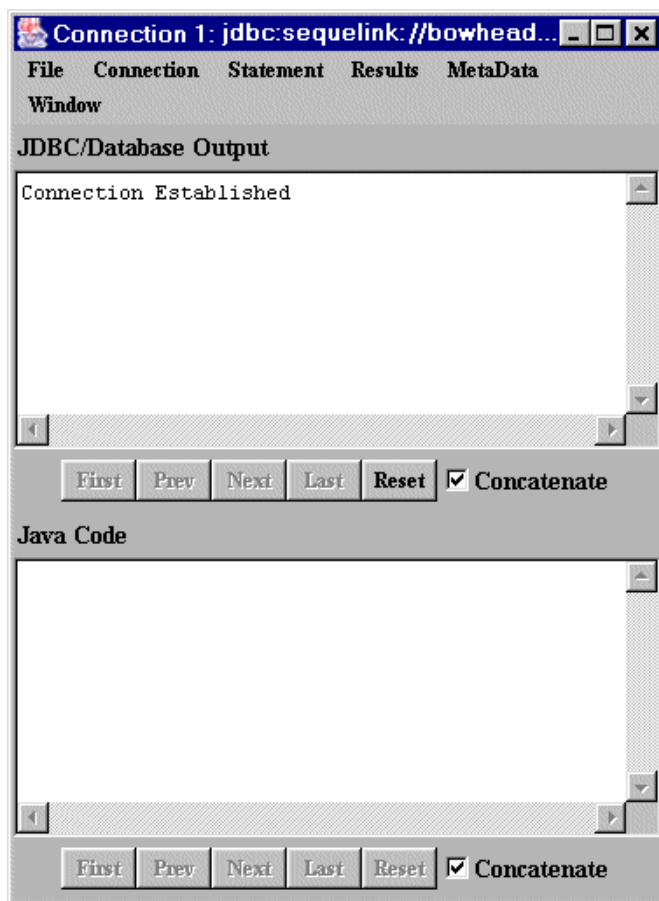
Connecting Using a Data Source

- 1 From the DataDirect Test main window menu, select **Connection / Connect to DB via Data Source**. DataDirect Test displays the Select A Datasource window.



- 2 Select a data source from the Defined Datasources pane. In the User Name and Password fields, type the required user and password connection properties; then, click **Connect**. See [Chapter 8 “Developing JDBC Applications” on page 327](#) for information about JDBC connection properties.

- 3 If the connection was successful, the Connection window appears and displays `Connection Established` in the JDBC/Database Output scroll box.



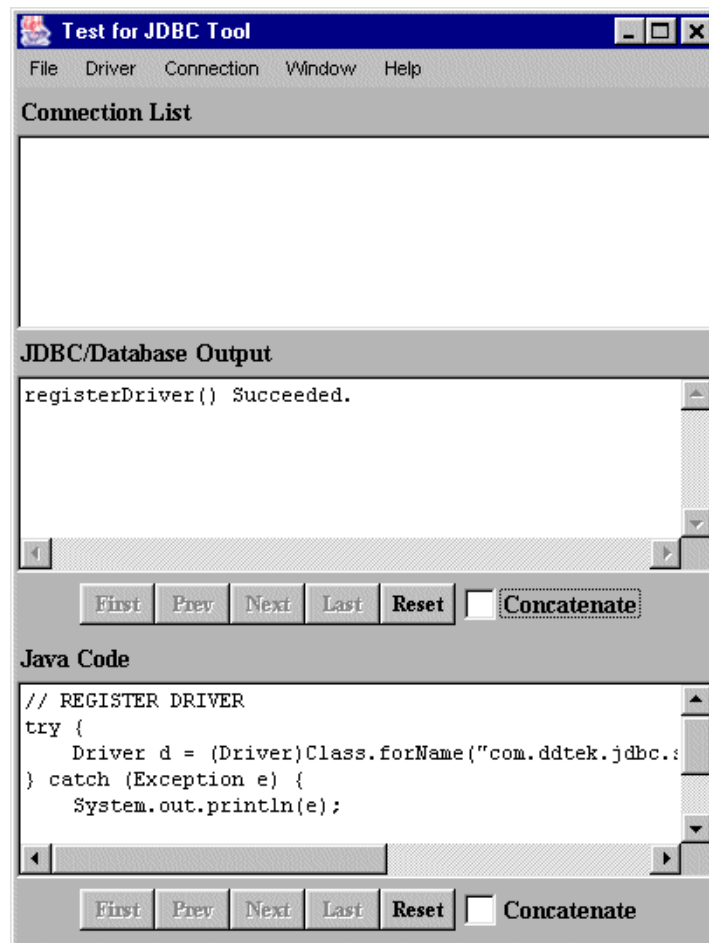
Connecting Using Driver/Database Selection

- 1 From the DataDirect Test main window menu, select **Driver / Register Driver**. DataDirect Test prompts you for a JDBC driver name.

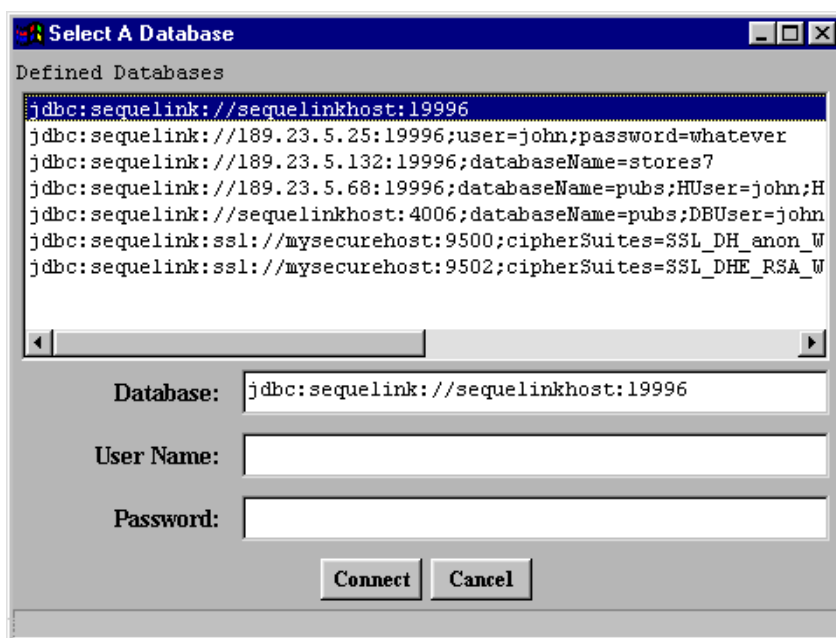
- 2 In the Please Supply a Driver URL field, make sure that a driver is specified, as in the following example; then, click **OK**.

```
com.ddtek.jdbc.sequelink.SequeLinkDriver
```

If the JDBC driver was registered successfully, the main DataDirect Test window appears with a confirmation in the JDBC/Database Output scroll box.

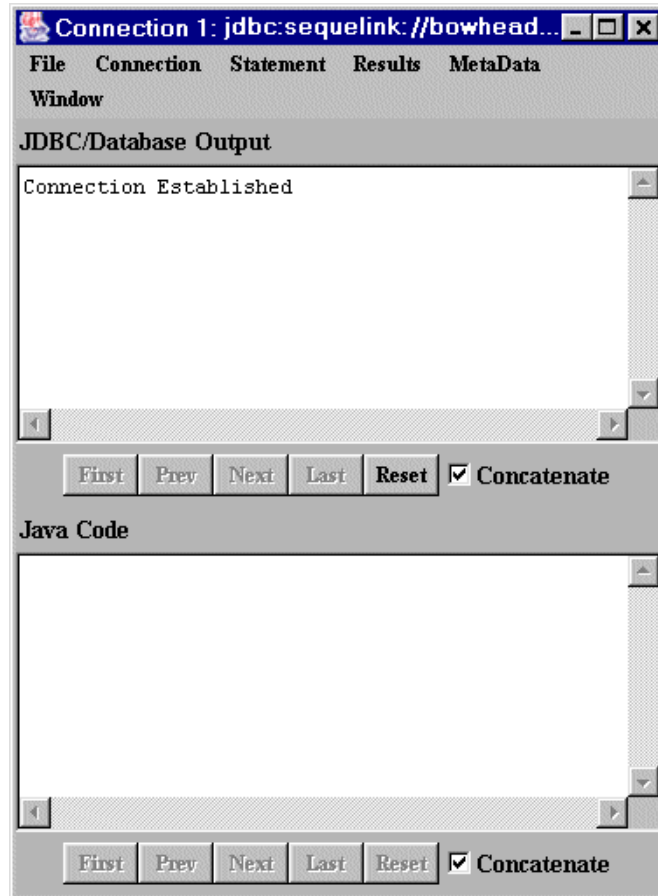


- 3 Select **Connection / Connect to DB** from the main menu. JDBC prompts with a list of default connection URLs.



- 4 Select one of the default JDBC driver connection URLs. In the Database field, modify the default values of the connection URL appropriately for your environment.
- 5 In the User Name and Password fields, type the required user and password connection properties; then, click **Connect**. See [Chapter 8 "Developing JDBC Applications" on page 327](#) for information about JDBC connection properties.

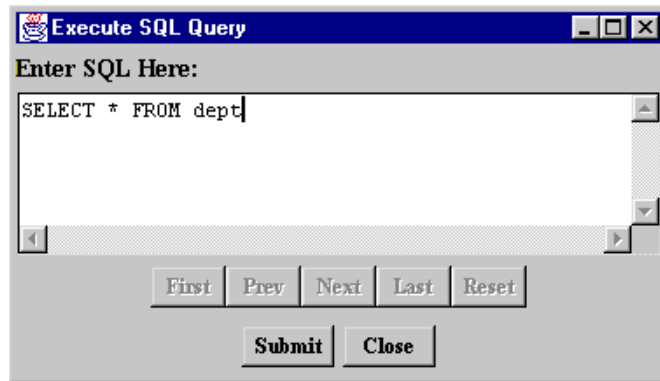
- 6 If the connection was successful, the Connection window appears and displays `Connection Established` in the JDBC/Database Output scroll box.



Executing a Simple Select Statement

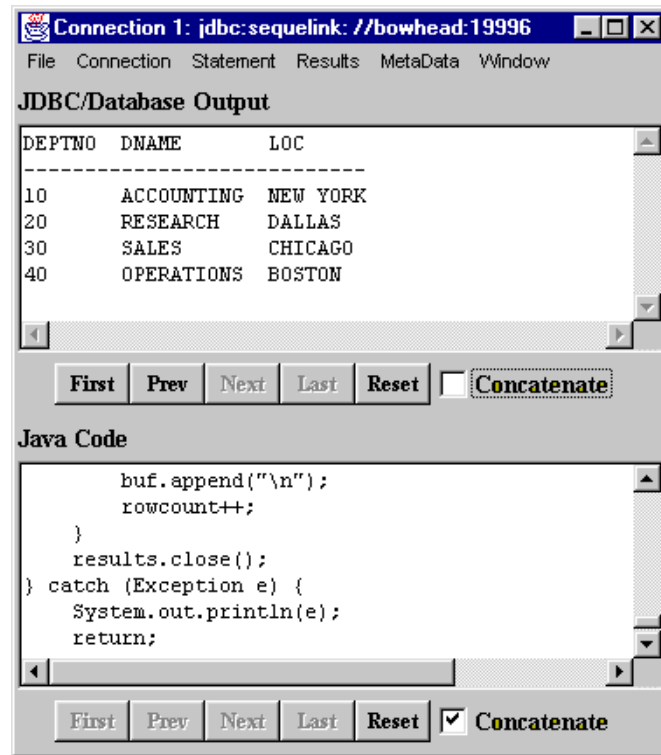
This example explains how to execute a simple Select statement and retrieve the results.

- 1 From the Connection window menu, select **Connection / Create Statement**. The connection window indicates that the creation of the statement was successful.
- 2 Select **Statement / Execute Stmt Query**. DataDirect Test displays a dialog box that prompts for a SQL statement.
- 3 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 4 Select **Results / Show All Results**. The data from your result set is displayed.

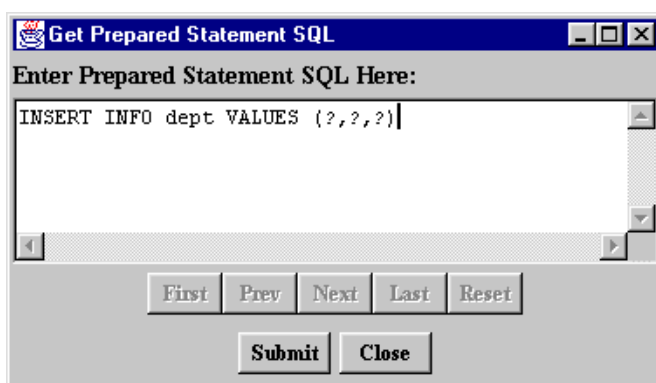


- 5 Scroll through the code in the Java Code scroll box to see which JDBC calls have been implemented by DataDirect Test.

Executing a Prepared Statement

This example explains how to execute a parameterized statement multiple times.

- 1 From the Connection window menu, select **Connection / Create Prepared Statement**. DataDirect Test prompts you for a SQL statement.
- 2 Specify the Insert statement that you want to execute.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Set Prepared Parameters**. To set the value and type for each parameter:
 - a Type the parameter number.
 - b Select the parameter type.
 - c Type the parameter value.
 - d Click **Set** to pass this information to the JDBC driver.

#	Type	Value
1	short	50
2	String	DEVELOPMENT
3	String	SAN FRANCISCO

Parameter #:

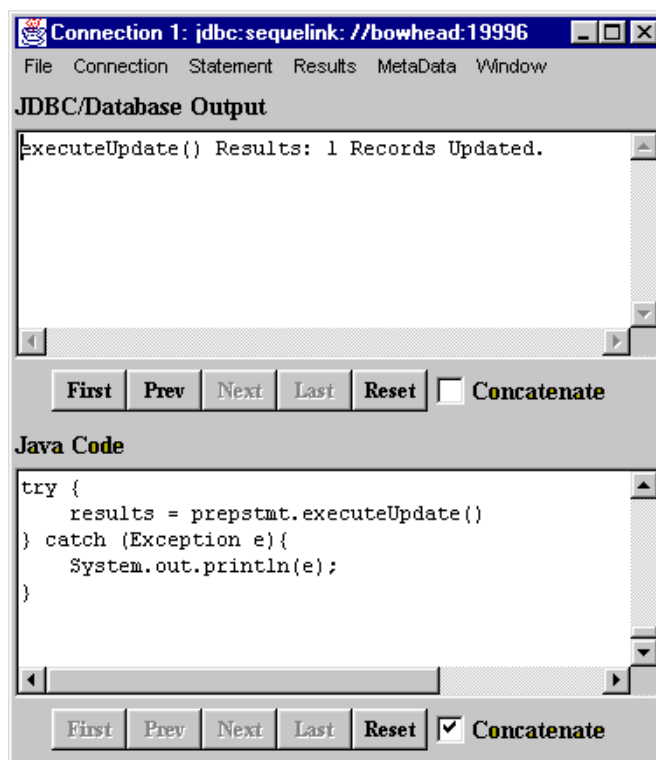
Parameter Type:

Use Calendar: ☐ Zone:

Parameter Value:

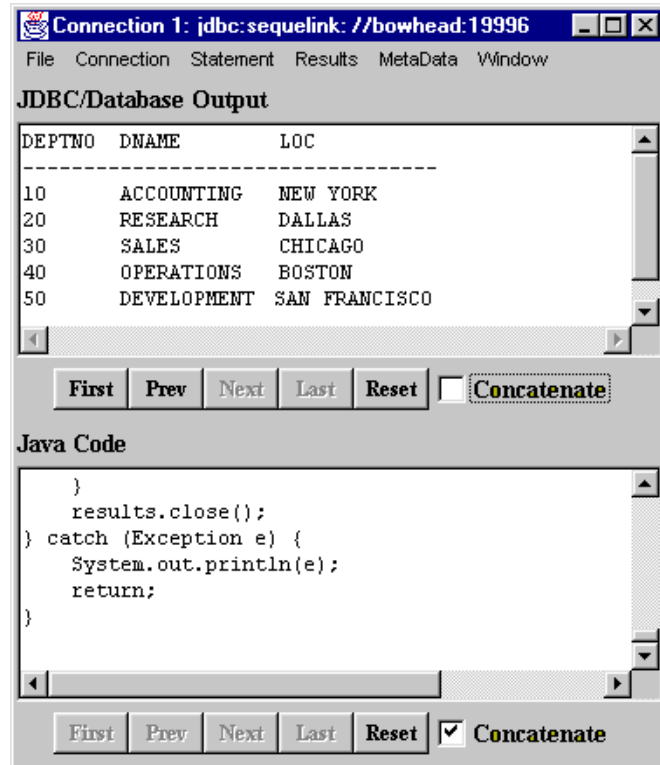
- 4 When you are finished, click **Close**.

- 5 Select **Statement / Execute Stmt Update**. The JDBC/Database Output scroll box indicates that one row has been inserted.



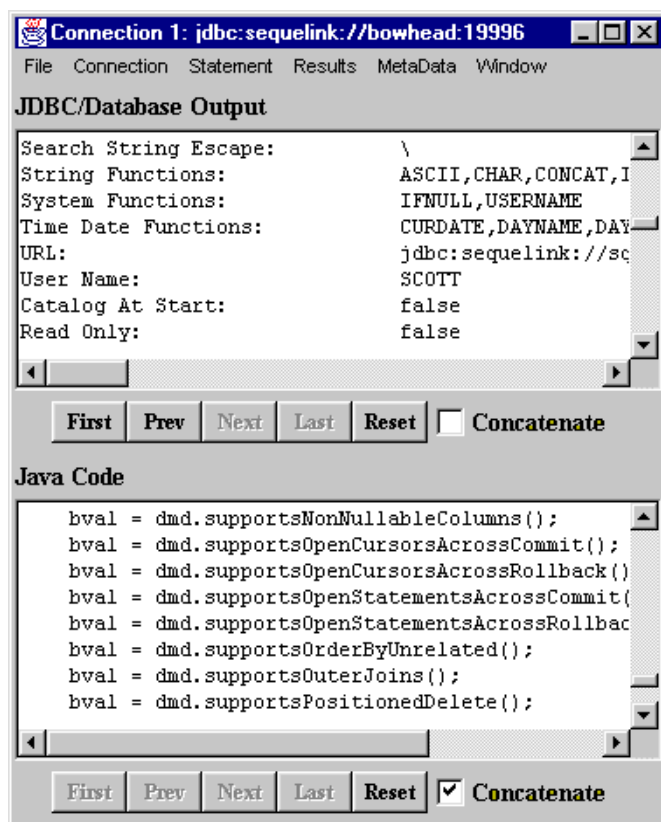
- 6 If you want to insert multiple records, repeat [Step 3](#) and [Step 5](#) for each record.

- 7 If you repeat the steps described in [“Executing a Simple Select Statement”](#) on page 272, you will see that the previously inserted records are also returned.



Retrieving Database Metadata

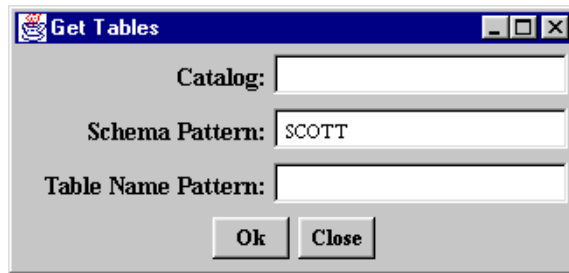
- 1 From the Connection window menu, select **Connection / Get DB Meta Data**.
- 2 Select **MetaData / Show Meta Data**. Information about the JDBC driver and the database to which you are connected is returned.



- 3 Scroll through the Java code in the Java Code scroll box to find out which JDBC calls have been implemented by DataDirect Test.

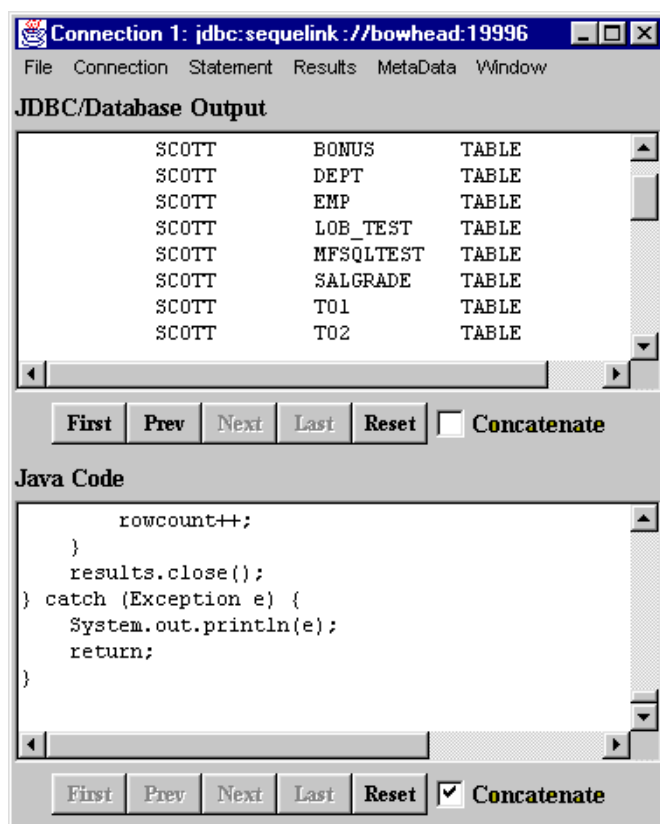
Metadata also allows you to query the database catalog (enumerate the tables in the database, for example). In this example, we will query all tables that are owned by the user SCOTT.

- 4 Select **MetaData / Tables**.
- 5 In the Schema Pattern field, type SCOTT.



- 6 Click **Ok**. The Connection window indicates that getTables() succeeded.

- 7 Select **Results / Show All Results**. All tables owned by SCOTT are returned.



Scrolling Through a Result Set

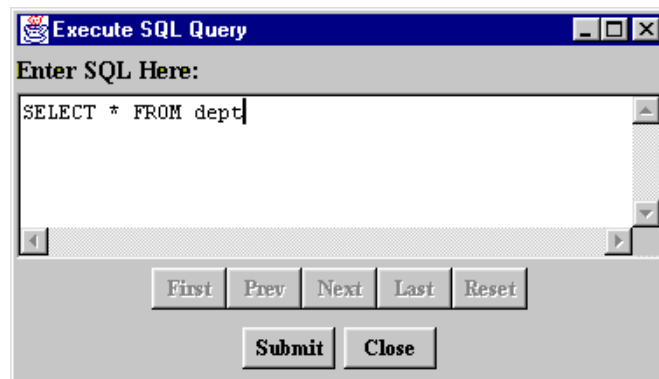
NOTE: Scrollable result sets are supported by JDBC 2.0 and higher and require a Java 2 Platform (J2SE 1.4 or higher)-compatible Java Virtual Machine.

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**. DataDirect Test prompts you for a result set type and concurrency.
- 2 In the resultSetType field, select **TYPE_SCROLL_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR_READ_ONLY**.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 5 Select **Results / Scroll Results**. The Scroll Result Set window indicates that the cursor is positioned before the first row.



- 6 Click the **Absolute**, **Relative**, **Before**, **First**, **Prev**, **Next**, **Last**, and **After** buttons as appropriate to navigate through the result set. After each action, the Scroll Result Set window displays the data at the current position of the cursor.

The screenshot shows a window titled "Scroll Result Set" with a table of data and navigation controls. The table has four columns: OPERATION, DEP, DNAME, and LOC. The data is as follows:

OPERATION	DEP	DNAME	LOC
next:1	10	ACCOUNTING	NEW YORK
next:2	20	RESEARCH	DALLAS
first:1	10	ACCOUNTING	NEW YORK
last:5	50	DEVELOPMENT	SAN FRANCISCO
previous:5	50	DEVELOPMENT	SAN FRANCISCO
previous:4	40	OPERATIONS	BOSTON
previous:3	30	SALES	CHICAGO

Below the table, the "Current Row" is displayed as 3. At the bottom, there are several buttons: "Absolute", "Relative", "Before", "First", "Prev", "Next", "Last", "After", and "Close". The "Prev" button is currently selected.

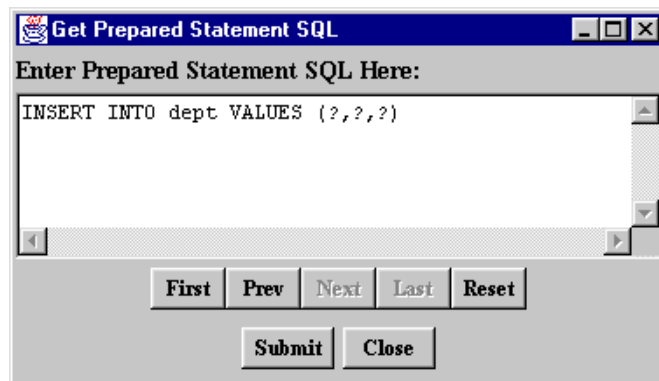
- 7 Click **Close**.

Batch Execution on a Prepared Statement

Batch execution on a prepared statement allows you to update or insert multiple records simultaneously. In some cases, this can significantly improve system performance because fewer round-trips to the database are required.

NOTE: Batch execution on a prepared statement is supported by the JDBC 2.0 and higher specifications and requires a Java 2 Platform (J2SE 1.4 or higher)-compatible Java Virtual Machine.

- 1 From the Connection window menu, select **Connection / Create Prepared Statement**.
- 2 Specify the Insert statement that you want to execute.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Add Stmt Batch**.

- 4 For each parameter:
 - a Type the parameter number.
 - b Select the parameter type.
 - c Type the parameter value.
 - d Click **Set**.

#	Type	Value
1	short	60
2	String	MARKETING
3	String	NEW YORK

Parameter #:

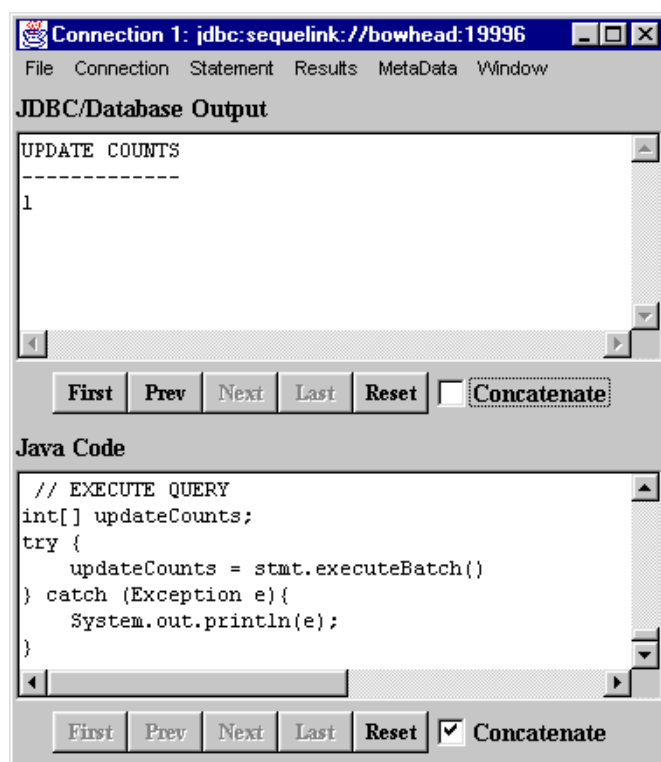
Parameter Type:

Use Calendar: ☐ Zone:

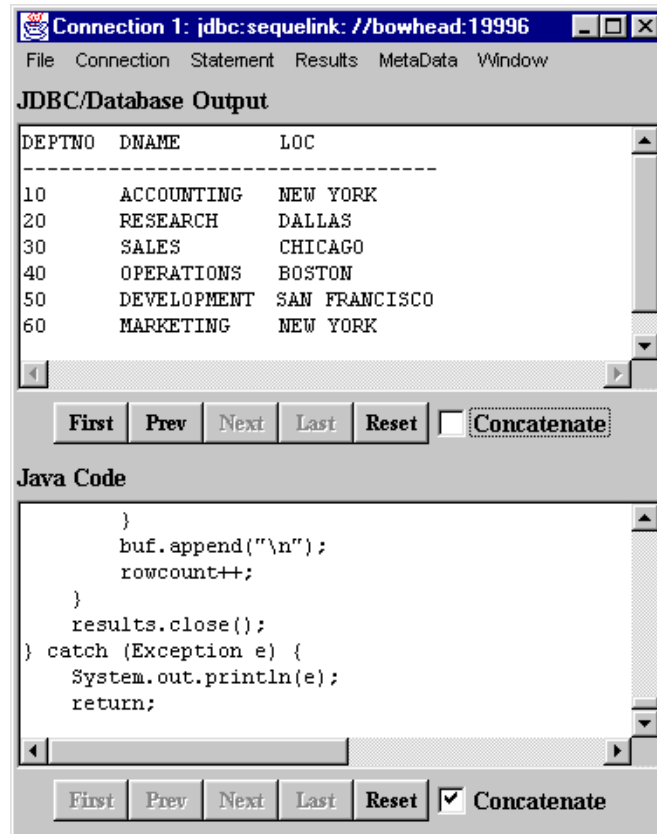
Parameter Value:

- 5 Click **Add** to add the specified set of parameters to the batch. To add multiple parameter sets to the batch, repeat [Step 3](#) through Step 5 as many times as necessary. When you are finished adding parameter sets to the batch, click **Close**.

- 6 Select **Statement / Execute Stmt Batch**. DataDirect Test displays the rowcount for each of the elements in the batch.



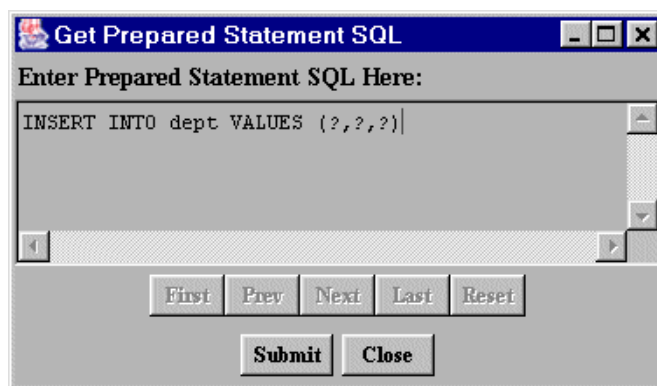
- 7 If you re-execute the Select statement from [“Executing a Simple Select Statement” on page 272](#), you see that the previously inserted records are returned.



Returning ParameterMetaData

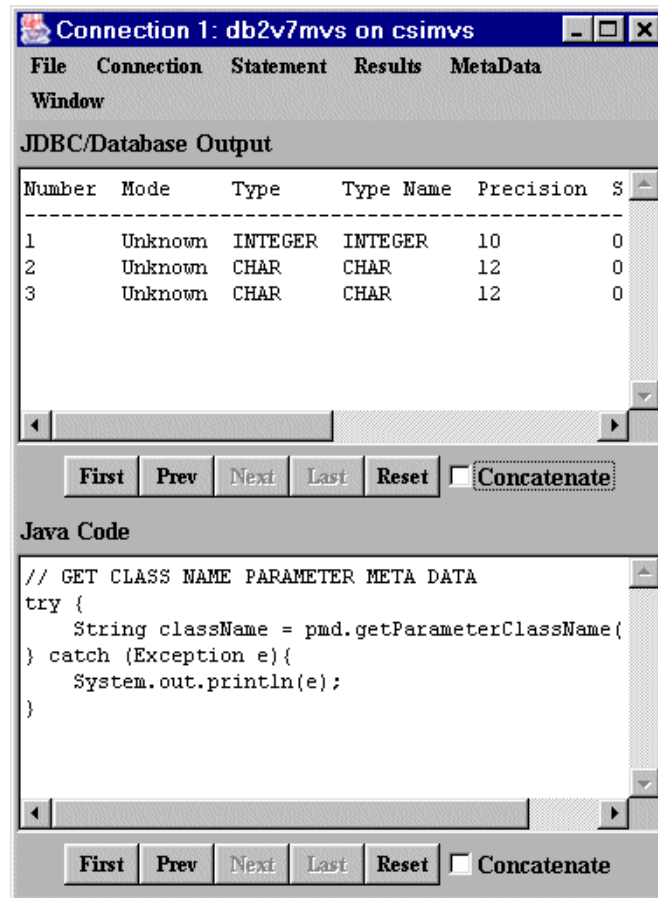
NOTE: Returning ParameterMetaData is a JDBC 3.0 feature and requires a J2SE 1.4 or higher Java Virtual Machine.

- 1 From the Connection window menu, select **Connection / Create Prepared Statement**.
- 2 Specify the prepared statement that you want to execute.



Click **Submit**; then, click **Close**.

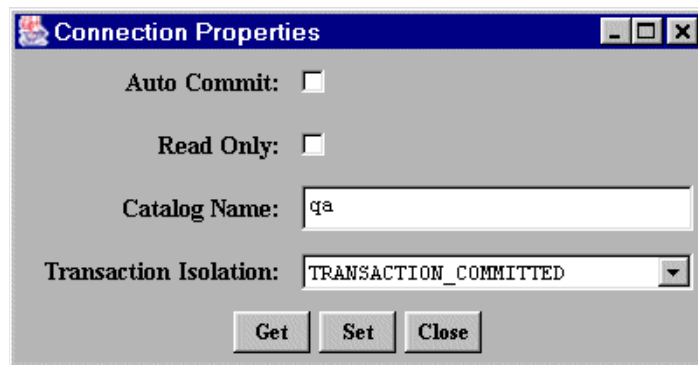
- 3 Select **Statement / Get ParameterMetaData**. The Connection window displays ParameterMetaData.



Establishing Savepoints

NOTE: Savepoints is a JDBC 3.0 feature and requires a J2SE 1.4 or higher Java Virtual Machine.

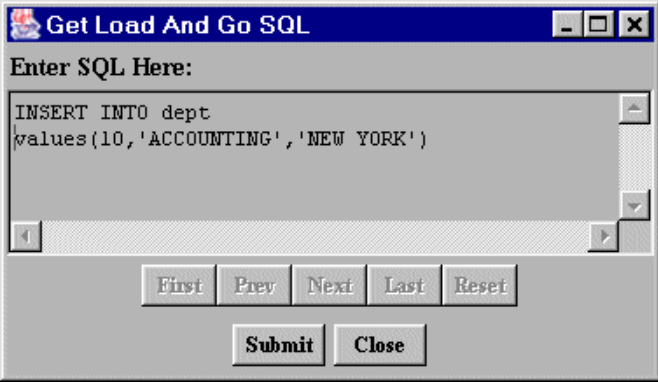
- 1 From the Connection window menu, select **Connection / Connection Properties**.
- 2 Select **TRANSACTION_COMMITTED** from the Transaction Isolation drop-down list. Do not select the Auto Commit check box.



Click **Set**; then, click **Close**.

- 3 From the Connection window menu, select **Connection / Load and Go**. The Get Load And Go SQL window appears.

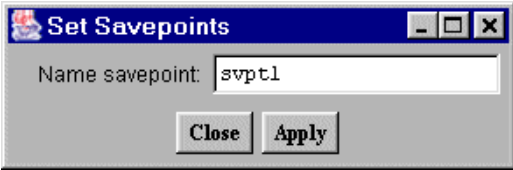
- 4 Specify the statement that you want to execute.



The dialog box titled "Get Load And Go SQL" has a text area labeled "Enter SQL Here:" containing the SQL statement: `INSERT INTO dept values(10,'ACCOUNTING','NEW YORK')`. Below the text area are five buttons: "First", "Prev", "Next", "Last", and "Reset". At the bottom are two buttons: "Submit" and "Close".

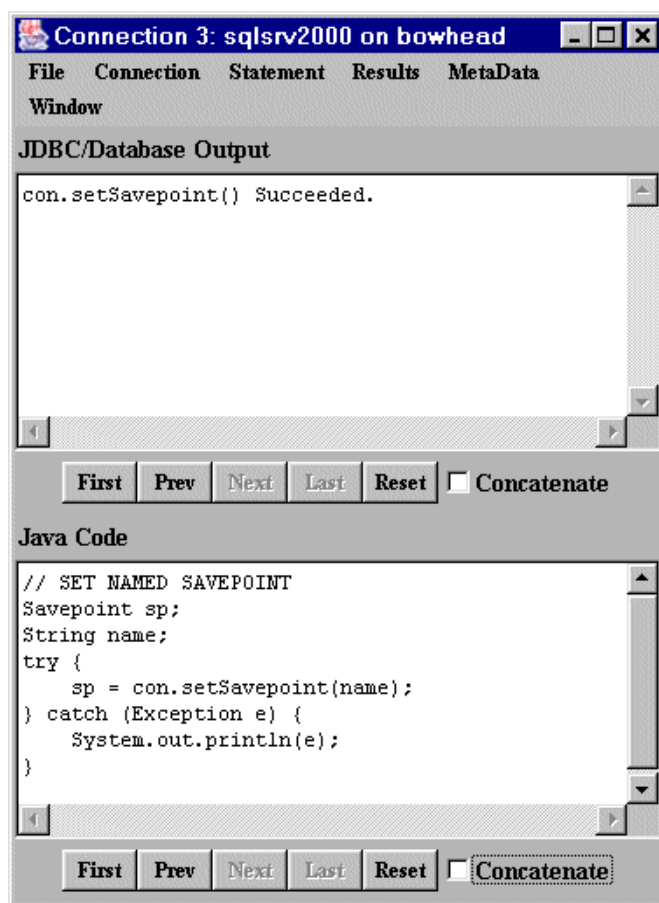
Click **Submit**.

- 5 Select **Connection / Set Savepoint**. In the Set Savepoints window, specify a savepoint name.

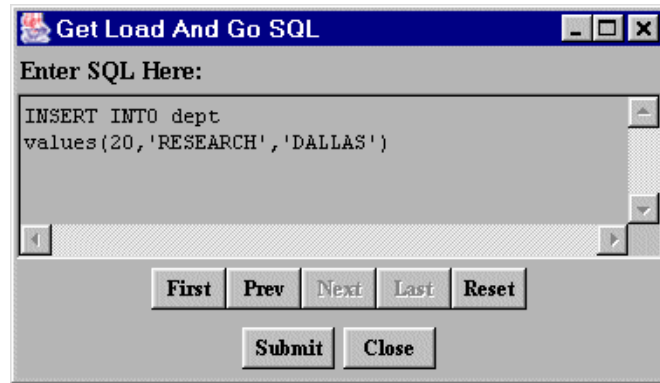


The dialog box titled "Set Savepoints" has a text field labeled "Name savepoint:" with the value "svpt1" entered. Below the text field are two buttons: "Close" and "Apply".

Click **Apply**; then, click **Close**. The Connection window indicates whether or not the savepoint succeeded.

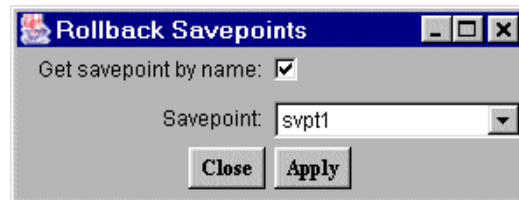


- 6 Return to the Get Load And Go SQL window and specify another statement.

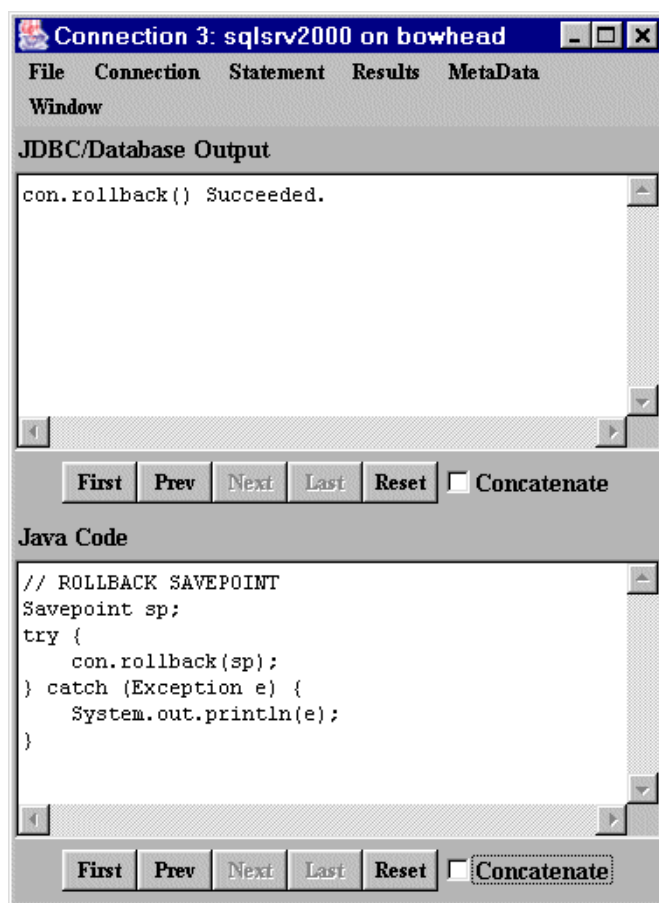


Click **Submit**.

- 7 Select **Connection / Rollback Savepoint**. In the Rollback Savepoints window, specify the savepoint name.



Click **Apply**; then, click **Close**. The Connection window indicates whether or not the savepoint rollback succeeded.

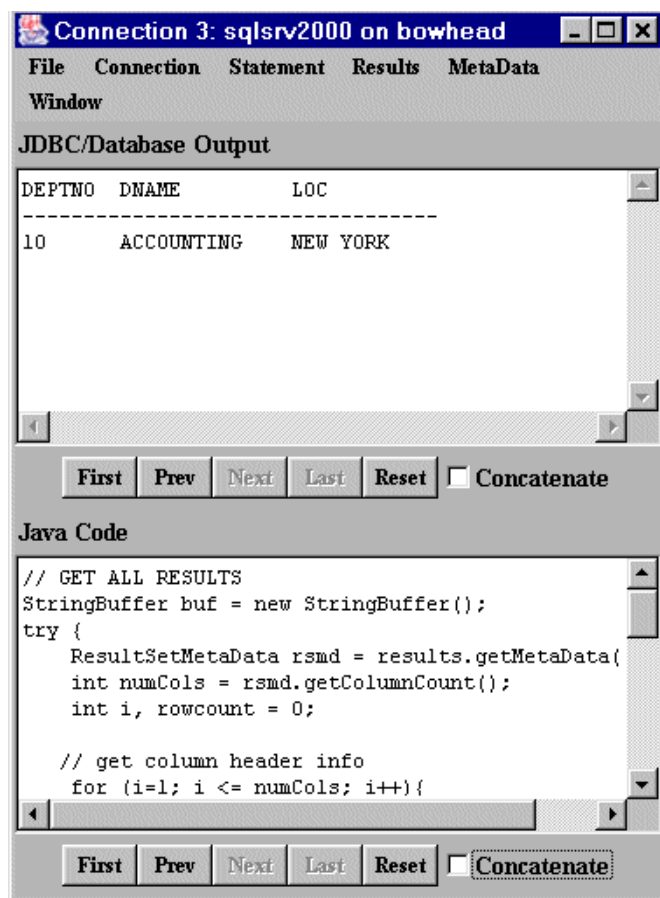


- 8 Return to the Get Load And Go SQL window and specify another statement.



Click **Submit**; then, click **Close**.

The Connection window displays data that was inserted before the first Savepoint. The second insert was rolled back.

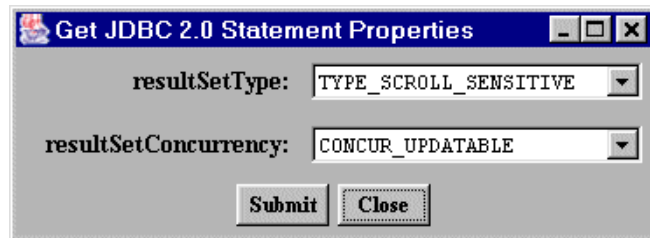


Updatable Result Sets

The following examples illustrate Updatable result sets by deleting, inserting, and updating a row.

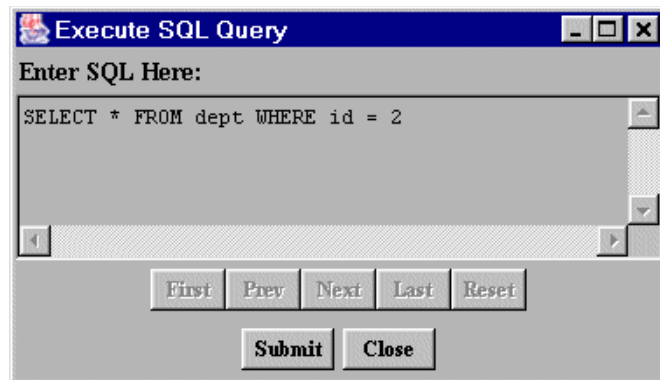
Deleting a Row

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**.
- 2 In the resultSetType field, select **TYPE_SCROLL_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR_UPDATABLE**.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 5 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.

The screenshot shows the 'Inspect Result Set' window. At the top is a table with the following data:

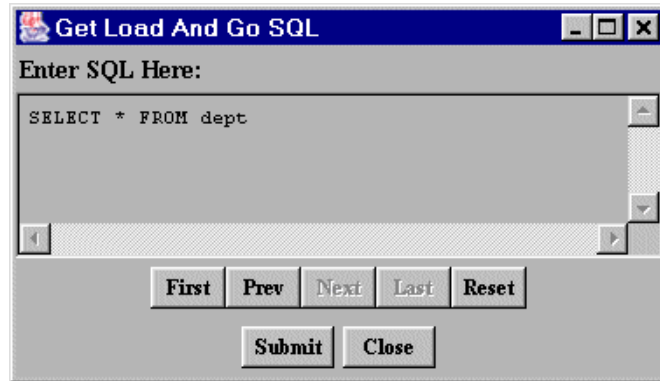
Col#	Name	Type
1	DEPTNO	int
2	DNAME	char
3	LOC	char
4	id	numeric identity

Below the table, the 'Current Row' is set to 1. The 'Retrieve By Column' dropdown is set to 'Index'. The 'Data Type' dropdown is set to 'String', and the 'Use Scale' checkbox is unchecked with a value of 0. The 'Use Calendar' checkbox is unchecked, and the 'Zone' dropdown is set to 'Etc/GMT+12'. There are two text input fields for 'Get Cell Value' and 'Set Cell Value', each with a small grid icon to its right. The 'Auto Traverse' checkbox is checked. Below this, there are buttons for 'Absolute' and 'Relative', with a text input field containing '1' between them. A row of navigation buttons includes 'Before', 'First', 'Prev', 'Next' (which is highlighted with a dotted border), 'Last', and 'After'. Below these are 'Get Cell' and 'Set Cell' buttons. At the bottom, there are five buttons: 'Update Row', 'Delete Row', 'Insert Row', 'Move to insert row', and 'Move to current row'. A 'Close' button is located at the very bottom center.

- 6 Click **Next**. Current Row changes to 1.

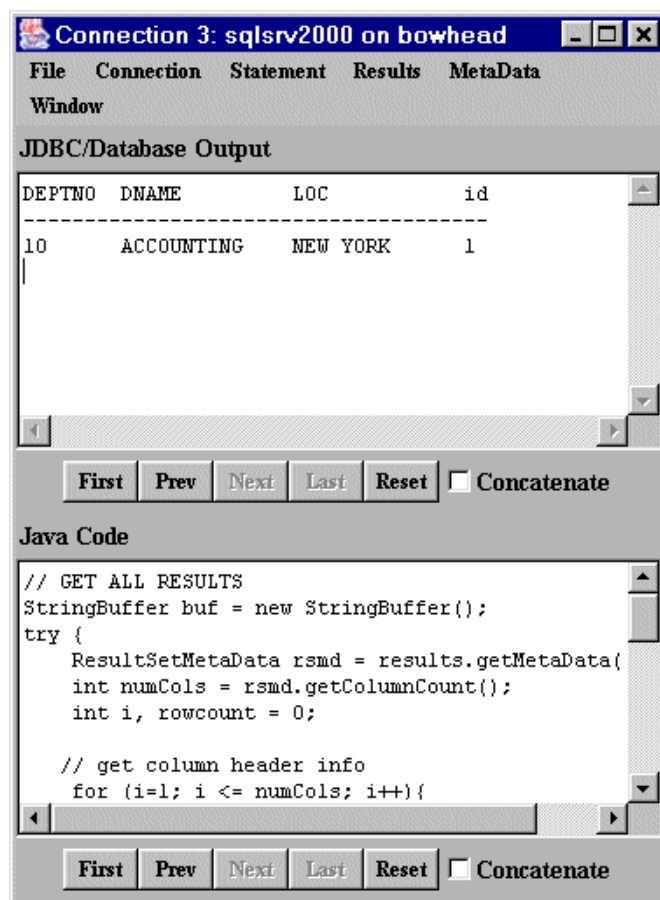
- 7 Click **Delete Row**.

- 8 To verify the result, return to the Connection menu and select **Connection / Load And Go**. The Get Load and Go SQL window appears.
- 9 Specify the statement that you want to execute.



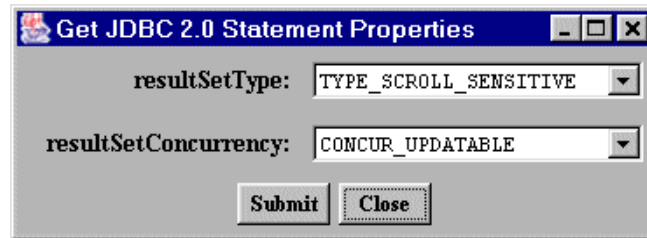
Click **Submit**; then, click **Close**.

10 The Connection window shows one row returned.



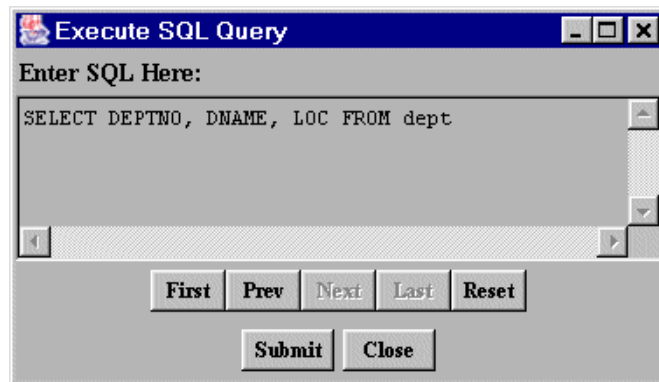
Inserting a Row

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**.
- 2 In the resultSetType field, select **TYPE_SCROLL_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR_UPDATABLE**.



Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 5 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.

Inspect Result Set

Col#	Name	Type
1	DEPTNO	int
2	DNAME	char
3	LOC	char

Current Row:

Insert row

Retrieve By Column:

Index

Data Type:

String

Use Scale:

☐

0

Use Calendar:

☐

Zone:

Etc/GMT+12

Get Cell Value:

Set Cell Value:

DALLAS

Auto Traverse:

☒

Absolute

1

Relative

Before

First

Prev

Next

Last

After

Get Cell

Set Cell

Update Row

Delete Row

Insert Row

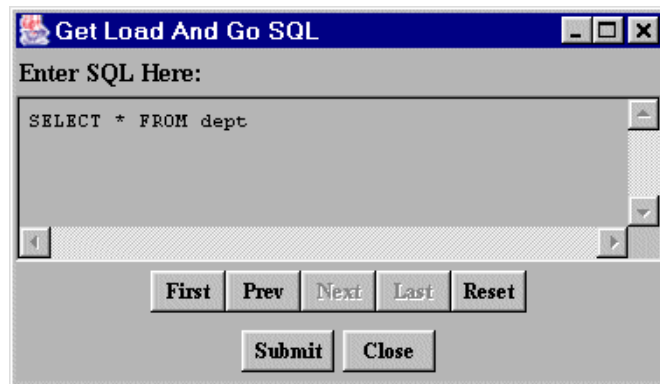
Move to insert row

Move to current row

Close

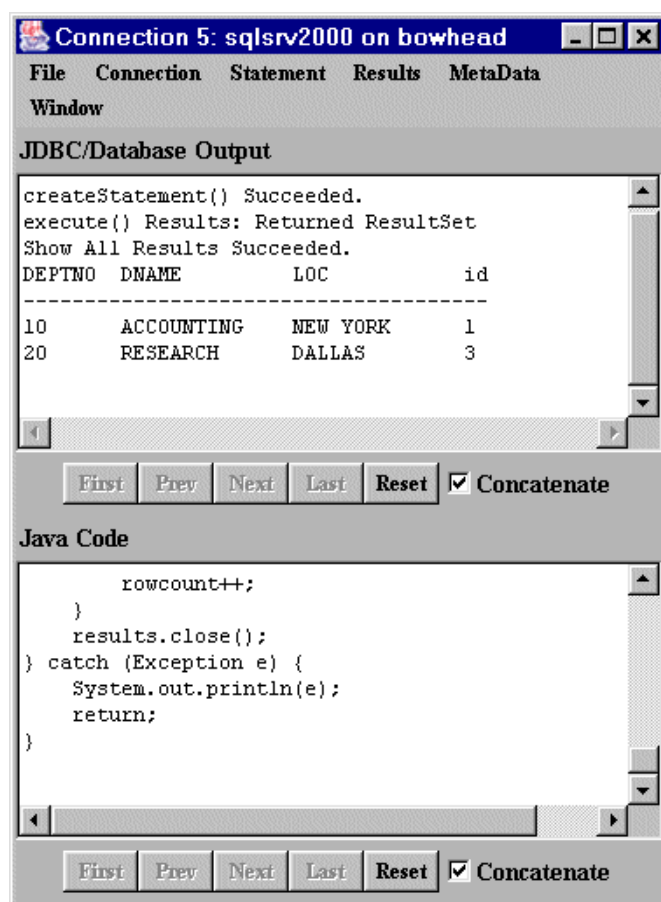
- 6 Click **Move to insert row**; Current Row is now Insert row.

- 7 Change Data Type to int. In Set Cell Value, enter 20. Click **Set Cell**.
- 8 Select the second row in the top pane. Change the Data Type to String. In Set Cell Value, enter RESEARCH. Click **Set Cell**.
- 9 Select the third row in the top pane. In Set Cell Value, enter DALLAS. Click **Set Cell**.
- 10 Click **Insert Row**.
- 11 To verify the result, return to the Connection menu and select **Connection / Load And Go**. The Get Load and Go SQL window appears.
- 12 Specify the statement that you want to execute.



Click **Submit**; then, click **Close**.

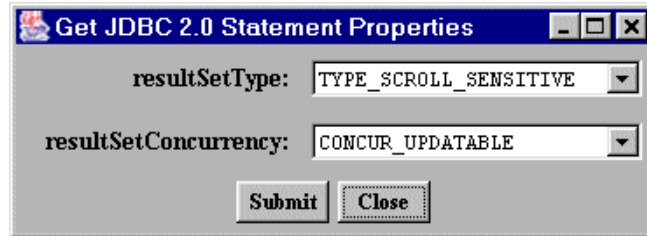
13 The Connection window shows two rows returned.



Note that the ID will be 3 for the row just inserted, because it is an auto increment column.

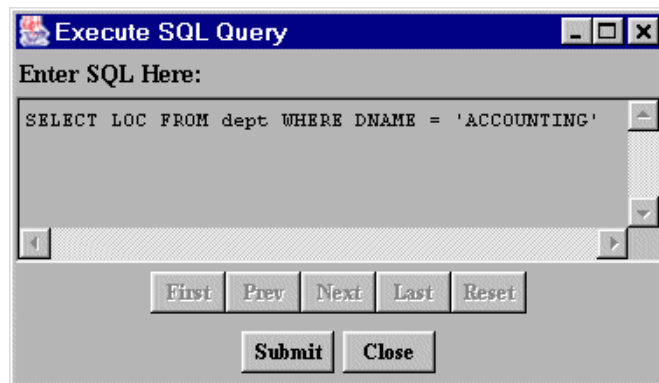
Updating a Row

- 1 From the Connection window menu, select **Connection / Create JDBC 2.0 Statement**.
- 2 In the resultSetType field, select **TYPE_SCROLL_SENSITIVE**. In the resultSetConcurrency field, select **CONCUR_UPDATABLE**.



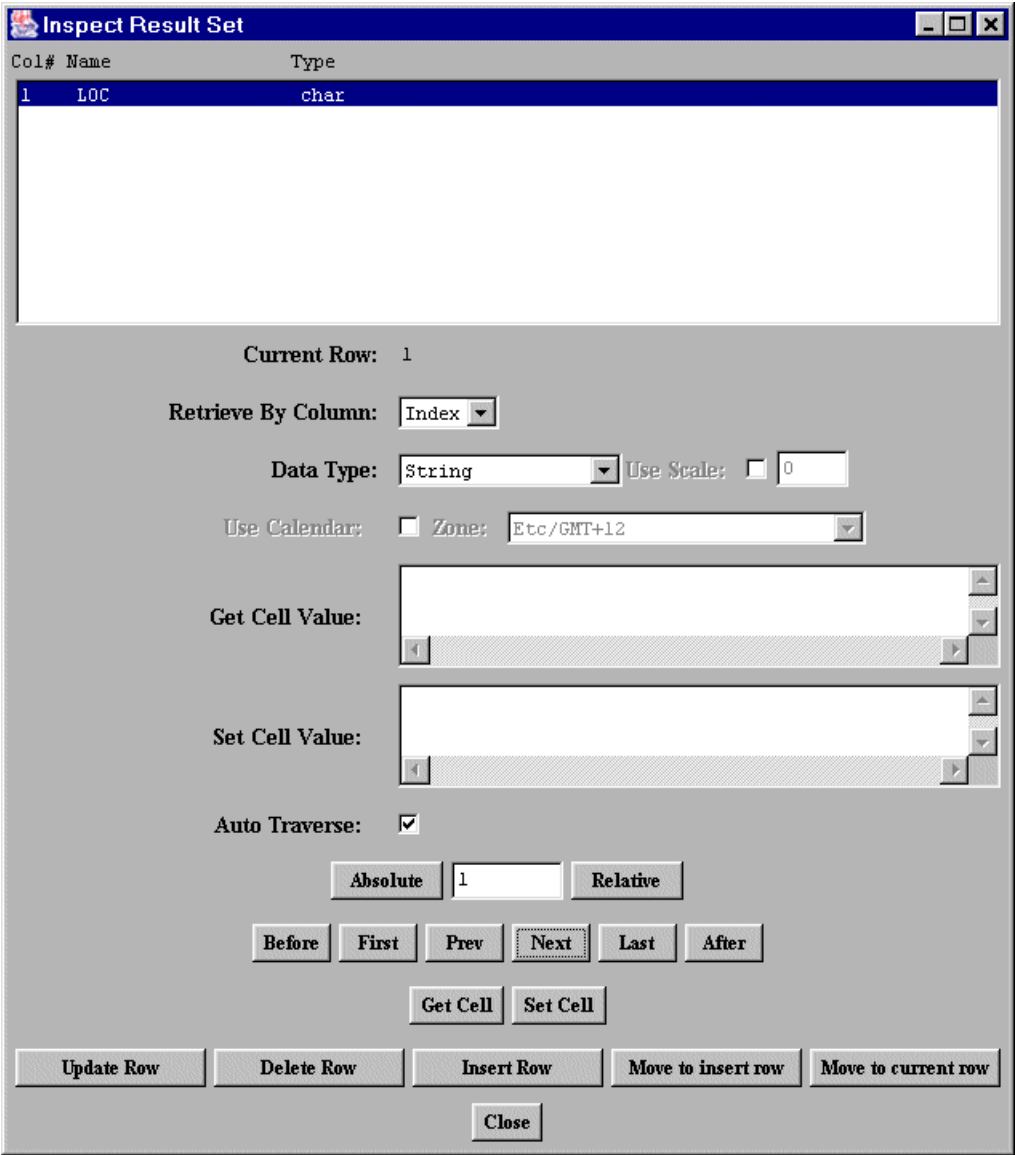
Click **Submit**; then, click **Close**.

- 3 Select **Statement / Execute Stmt Query**.
- 4 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 5 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.



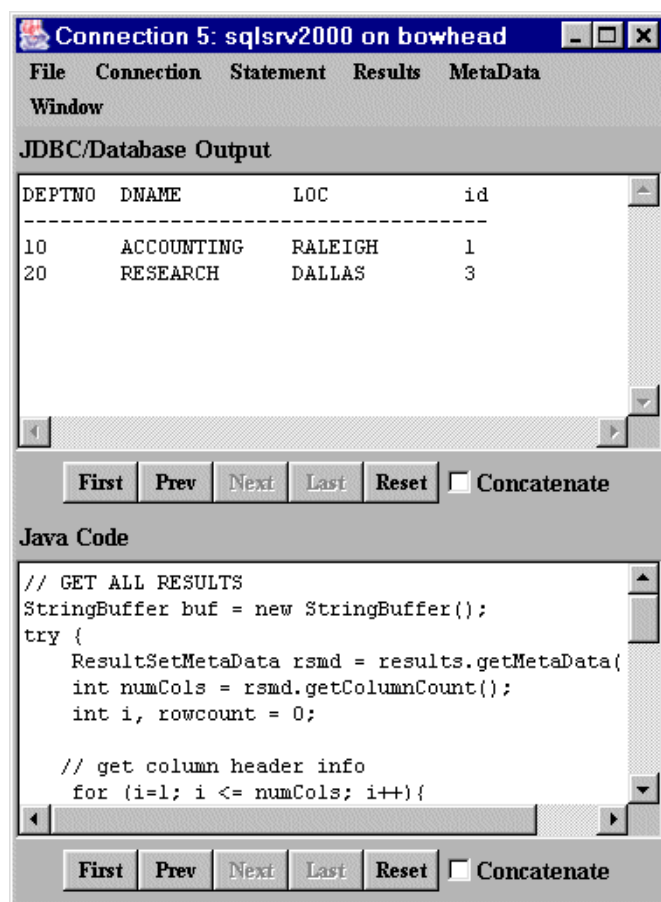
- 6 Click **Next**. Current Row changes to 1.
- 7 In Set Cell Value, enter `RALEIGH`. Click **Set Cell**.

- 8 Click **Update Row**.
- 9 To verify the result, return to the Connection menu and select **Connection / Load And Go**. The Get Load and Go SQL window appears.
- 10 Specify the statement that you want to execute.



Click **Submit**; then, click **Close**.

- 11 The Connection window shows LOC for accounting changed from NEW YORK to RALEIGH.

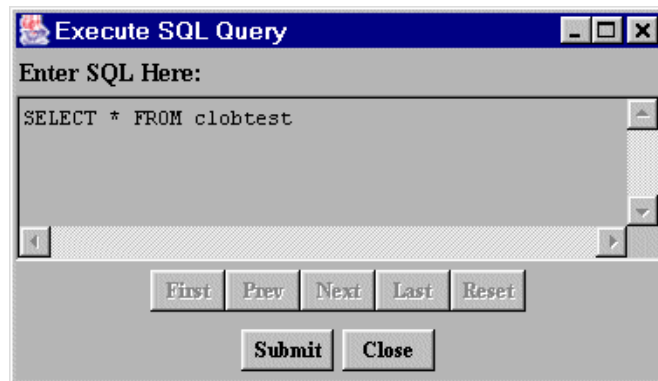


LOB Support

NOTE: LOB support (Blobs and Clobs) is a JDBC 3.0 feature and requires a J2SE 1.4 or higher Java Virtual Machine.

The following example uses CLOB data; however, this procedure also applies to BLOB data. This example illustrates only one of several ways in which LOB data can be processed.

- 1 From the Connection window menu, select **Connection / Create Statement**.
- 2 Select **Statement / Execute Stmt Query**.
- 3 Specify the Select statement that you want to execute.



Click **Submit**; then, click **Close**.

- 4 Select **Results / Inspect Results**. The Inspect Result Set window is displayed.

5 Click **Next**. Current Row changes to 1.

Inspect Result Set

Col#	Name	Type
1	CLOBCOL	clob

Current Row: 1

Retrieve By Column:

Index

Data Type:

String

Use Scale:

☐

0

Use Calendar:

☐

Zone:

Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse:

☒

Absolute

1

Relative

Before

First

Prev

Next

Last

After

Get Cell

Set Cell

Update Row

Delete Row

Insert Row

Move to insert row

Move to current row

Close

6 Deselect **Auto Traverse**. This disables automatic traversal to the next row.

7 Click **Get Cell**.

Inspect Result Set

Col#	Name	Type
1	CLOBCOL	clob

Current Row: 1

Retrieve By Column:

Data Type: Use Scale: ☐ 0

Use Calendar: ☐ Zone:

Get Cell Value (1,1):

Set Cell Value:

Auto Traverse: ☐

8 Values are returned in the Get Cell Value field.

9 Change the Data Type to Clob.

Inspect Result Set

Col#	Name	Type
1	CLOBCOL	clob

Current Row: 1

Retrieve By Column:

Index

Data Type:

Clob

Use Scale:

☐

0

Use Calendar:

☐

Zone:

Etc/GMT+12

Get Cell Value:

Set Cell Value:

Auto Traverse:

☒

Absolute

1

Relative

Before

First

Prev

Next

Last

After

Get Cell

Set Cell

Update Row

Delete Row

Insert Row

Move to insert row

Move to current row

Close

10 Click **Get Cell**. The Clob data window appears.

Clob data

Length: 1134

Position: 1

Length: 1134

Return Type: Character Stream

Cell Value:

Pattern:

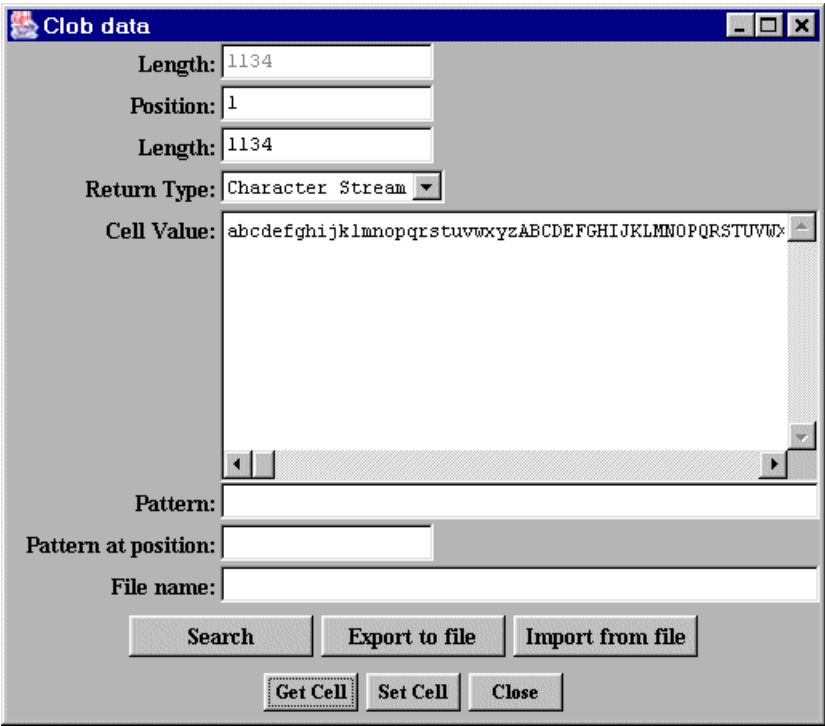
Pattern at position:

File name:

Search Export to file Import from file

Get Cell Set Cell Close

11 Click **Get Cell**.



12 Values are returned in the Cell Value field.

7 Tracking JDBC Calls

This chapter introduces DataDirect Spy, a development software component that allows you to track JDBC calls, and describes how to use it.

About DataDirect Spy™

DataDirect Spy is a software component for tracking JDBC calls at runtime. It passes calls issued by an application to an underlying JDBC driver and logs detailed information about those calls. DataDirect Spy provides the following advantages:

- Logging is JDBC 3.0-compliant, including support for the JDBC 2.0 Optional Package.
- Logging is consistent, regardless of the JDBC driver used.
- All parameters and function results for JDBC calls can be logged.
- Logging can be enabled without changing the application.
- DataDirect Spy can only be used with the SequeLink *for* JDBC Driver and the DataDirect Connect® *for* JDBC drivers.

When you enable DataDirect Spy for a connection, you can customize DataDirect Spy logging for your needs by setting one or multiple options for DataDirect Spy. For example, you may want to direct logging to a local file on your machine.

Enabling DataDirect Spy™

You can enable DataDirect Spy for a connection using any of the following methods:

- Specifying the SpyAttributes connection property for connections using the JDBC Driver Manager. See [“Using the JDBC Driver Manager” on page 317](#) for instructions.
- Specifying DataDirect Spy attributes using a JDBC data source. See [“Using JDBC Data Sources” on page 318](#) for instructions.
- Specifying a DataDirect Spy-specific connection URL. See [“Using the DataDirect Spy™ URL” on page 320](#) for instructions. If you use this method to enable DataDirect Spy, you first must register the DataDirect Spy driver.

Using any of these methods, you can set one or multiple options to customize DataDirect Spy logging. See [“DataDirect Spy™ Attributes” on page 323](#) for a complete list of supported attributes.

Using the JDBC Driver Manager

The SpyAttributes connection property allows you to specify a semi-colon separated list of DataDirect Spy attributes (see [“DataDirect Spy™ Attributes” on page 323](#)). The format for the value of the SpyAttributes property is:

```
(spy_attribute[;spy_attribute]...)
```

where *spy_attribute* is any valid DataDirect Spy attribute. See [“DataDirect Spy™ Attributes” on page 323](#) for a list of supported attributes.

Example on Windows:

The following example uses the JDBC Driver Manager to connect to Oracle while enabling DataDirect Spy:

```
Class.forName("com.ddtek.jdbc.sequelink.SequeLinkDriver");
Connection conn = DriverManager.getConnection
("jdbc:sequelink://QANT:4003;user=TEST;password=secret;
SpyAttributes=(log=(file)C:\\temp\\spy.log;
linelimit=80;logTName=yes;timestamp=yes)");
```

Using this example, DataDirect Spy would load the SequeLink for JDBC driver and log all JDBC activity to the spy.log file located in the C:\temp directory (log=(file)C:\temp\spy.log). The spy.log file logs a maximum of 80 characters on each line (linelimit=80) and includes the name of the current thread (logTName=yes) and a timestamp on each line in the log (timestamp=yes).

NOTE: If coding a path on Windows to the log file in a Java string, the backslash character (\) must be preceded by the Java escape character, a backslash. For example: log=(file)C:\\temp\\spy.log.

Example on Linux and UNIX:

The following code example uses the JDBC Driver Manager to connect to DB2 while enabling DataDirect Spy:

```
Class.forName("com.ddtek.jdbc.sequelink.SequeLinkDriver");
Connection conn = DriverManager.getConnection
    ("jdbc:sequelink://user=TEST;password=secret;
SpyAttributes=(log=(file)C:\\temp\\spy.log;
    linelimit=80;logTName=yes;timestamp=yes)");
```

Using this example, DataDirect Spy would load the DataDirect SequeLink driver and log all JDBC activity to the spy.log file located in the /tmp directory (log=(file)/tmp/spy.log). The spy.log file includes the name of the current thread (logTName=yes) and a timestamp on each line in the log (timestamp=yes).

Using JDBC Data Sources

The SequeLink *for* JDBC driver implements the following JDBC features:

- JNDI for Naming Databases
- Connection Pooling
- JTA

You can use DataDirect Spy to track JDBC calls made by a running application with any of these features. The com.ddtek.jdbcx.sequelink.datasource.SequeLinkDataSource class supports setting a semi-colon-separated list of DataDirect Spy attributes (see [“DataDirect Spy™ Attributes” on page 323](#)).

See [“Configuring JDBC Data Sources” on page 225](#) for more information about configuring data sources.

Example on Windows:

The following example creates a JDBC data source for the SequeLink *for* JDBC driver, which enables DataDirect Spy.

```
...
SequeLinkDataSource sds=new SequeLinkDataSource():
sds.setServerName("MyServer");
sds.setPortNumber(1234);
sds.setSpyAttributes("log=(file)C:\\tmp\\spy.log;logIS=yes;
logTName=yes");
Connection conn=sds.getConnection("scott","tiger");
...
```

Using this example, DataDirect Spy would load the SequeLink *for* JDBC driver and log all JDBC activity to the spy.log file located in the C:\\tmp directory (log=(file)C:\\tmp\\spy.log). In addition to regular JDBC activity, the spy.log file also logs activity on InputStream and Reader objects (logIS=yes). It also includes the name of the current thread (logTName=yes).

NOTE: If coding a path on Windows to the log file in a Java string, the backslash character (\\) must be preceded by the Java escape character, a backslash. For example: "log=

(file)C:\\temp\\spy.log;logIS=yes;logTName=yes".

Example on Linux and UNIX:

The following example creates a JDBC data source for the SequeLink *for* JDBC driver, which enables DataDirect Spy.

```
...
SequeLinkDataSource sds=new SequeLinkDataSource():
sds.setServerName("MyServer");
sds.setPortNumber(1234);
sds.setSpyAttributes("log=(file)C:\\tmp\\spy.log;logIS=yes;logTName=yes");
Connection conn=sds.getConnection("TEST","secret");
...
```

Using this example, DataDirect Spy would load the SequeLink *for* JDBC driver and log all JDBC activity to the `spy.log` file located in the `/tmp` directory (`log=(file)/tmp/spy.log`). The `spy.log` file includes the name of the current thread (`logTName=yes`).

Using the DataDirect Spy™ URL

If you enable DataDirect Spy using a DataDirect Spy URL, you first must register the DataDirect Spy driver with the JDBC Driver Manager. Registering a driver tells the Driver Manager which driver to load. The DataDirect Spy driver name is `com.ddtek.jdbc.SpyDriver`.

For example, to load the DataDirect Spy driver using the standard `Class.forName` method, include the following code in your application or applet and call `DriverManager.getConnection`:

```
Class.forName("com.ddtek.jdbcspy.SpyDriver");
```

For complete information about registering drivers with the JDBC Driver Manager, see [“Registering the JDBC Driver” on page 223](#).

You can enable DataDirect Spy using a DataDirect Spy-specific URL that has the following format:

```
jdbc:spy:{original-url};[key=value]...
```

where:

original-url is the connection URL of the SequeLink *for* JDBC driver. For example, the following connection URL uses the SequeLink *for* JDBC driver:

```
jdbc:sequelink://QANT:4003;databaseName=Oracle;  
OSUser=qauser;OSPassword=null12
```

key=value is a semi-colon separated list of DataDirect Spy attributes (see [“DataDirect Spy™ Attributes” on page 323](#)).

Example on Windows:

The following example uses a DataDirect Spy URL to enable DataDirect Spy:

```
jdbc:spy:{jdbc:sequelink://QANT:4003;databaseName=Oracle;  
OSUser=gauser;OSPassword=null12};log=System.out;  
linelimit=72
```

Using this example, DataDirect Spy would load the JDBC driver and log all JDBC activity to the Java output standard file, `System.out` (`log=System.out`). The `spy.log` file logs a maximum of 72 characters on each line (`linelimit=72`).

Example on Linux/UNIX:

The following example uses a DataDirect Spy URL to enable DataDirect Spy:

```
jdbc:spy:{jdbc:sequelink://QANT:4003;  
databaseName=Oracle;OSUser=gauser;OSPassword=null12};  
log=System.out;linelimit=72
```

Using this example, DataDirect Spy would load the JDBC driver and log all JDBC activity to the Java output standard file, `System.out` (`log=System.out`). The `spy.log` file logs a maximum of 72 characters on each line (`linelimit=72`).

Registering the DataDirect Spy™ JDBC Driver

To use the Spy JDBC Driver, you first must register it with the JDBC Driver Manager. You can register the Spy JDBC Driver in any of the following ways:

- **Method 1:** Set the Java property `jdbc.drivers` using the Java `-D` option. The `jdbc.drivers` property is defined as a colon-separated list of driver class names. For example:

```
com.ddtek.jdbcspy.SpyDriver:  
com.ddtek.jdbc.sequelink.SequeLinkDriver
```

The `jdbc.drivers` property can be set like other Java properties, using the `-D` option. For example:

```
java -Djdbc.drivers=com.ddtek.jdbcspy.SpyDriver
```

- **Method 2:** Set the Java property `jdbc.drivers` from within your Java application or applet. To do this, code the following lines in your Java application or applet, and call `DriverManager.getConnection()`:

```
Properties p = System.getProperties();  
p.put ("jdbc.drivers", "com.ddtek.jdbcspy.SpyDriver");  
System.setProperties (p);
```

- **Method 3:** Explicitly load the driver class using the standard `Class.forName()` method. To do this, code the following line and call `DriverManager.getConnection()`:

```
Class.forName ("com.ddtek.jdbcspy.SpyDriver");
```

DataDirect Spy™ Attributes

DataDirect Spy uses the following format as a connection URL:

```
jdbc:spy:{original-url};[key=value]...
```

where *original-url* is the connection URL of the underlying JDBC driver.

In addition, you can specify the following options:

<code>log=System.out</code>	Redirects logging to the Java output standard, System.out.
<code>log=(file)<i>filename</i></code>	Redirects logging to the file specified by <i>filename</i> .
<code>load=<i>classname</i></code>	<p>Loads the driver specified by <i>classname</i>. The default value is <code>com.ddtek.sequelink.jdbc.SequelinkDriver</code>.</p> <p>This attribute is supported only when you enable DataDirect Spy using the DataDirect Spy URL.</p>
<code>linelimit=<i>numberofchars</i></code>	<p>The maximum number of characters, specified by <i>numberofchars</i>, that Spy will log on one line.</p> <p>When set to no (the initial default), there is no maximum limit on the number of characters.</p>
<code>logLobs={yes no}</code>	Specifies whether Spy logs activity on Blob / Clob. The initial default is no.
<code>logIS={yes no nosingleread}</code>	<p>Specifies whether DataDirect Spy logs activity on InputStreams.</p> <p>When <code>logIS=nosingleread</code>, logging on InputStream and Reader objects is active; however, logging of the single-byte read <code>InputStream.read</code> or single-character <code>Reader.read</code> is suppressed to prevent generating large log files that contain single-byte or single character read messages.</p> <p>When set to no (the initial default), DataDirect Spy does not log activity on InputStreams.</p>

<code>logTName={yes no}</code>	<p>Specifies whether DataDirect Spy logs the name of the current thread.</p> <p>When set to no (the initial default), DataDirect Spy does not log the name of the current thread.</p>
<code>timestamp={yes no}</code>	<p>Specifies whether a timestamp should be included on each line of the DataDirect Spy log.</p> <p>When set to no (the initial default), DataDirect Spy does not include a timestamp on each line.</p>

Using DataDirect Spy™ with JDBC Data Sources

The JDBC driver implements the following features defined by the JDBC 2.0 Optional Package:

- JNDI for Naming Databases
- Connection Pooling
- Distributed Transaction Management (DTC)

You can use Spy to track JDBC calls with all of these features. The `com.ddtek.jdbcx.sequellink.SequelLinkDataSource` class supports the `SpyAttributes` connection attribute, which specifies a semi-colon-separated list of Spy attributes as described in [“DataDirect Spy™ Attributes” on page 323](#). See [“JDBC Connection URL Examples:” on page 225](#) for more information about configuring JDBC data sources.

The following examples create a `SequeLinkDataSource` and specifies that all JDBC calls must be logged in the file `/tmp/spy.log`, including the name of the current thread:

```
...
SequeLinkDataSource sds=new SequeLinkDataSource();
sds.setServerName("MyServer");
sds.setPortNumber(1234);
sds.setSpyAttributes("log=(file)/tmp/spy.log;logTName=yes");
Connection conn=sds.getConnection("scott","tiger");
...
```

Checking the DataDirect Spy™ Version

To check the version of your DataDirect Spy, change to the subdirectory containing DataDirect Spy (*install_dir/spy* where *install_dir* is your SequeLink installation directory). At a command prompt, enter the command:

On Windows:

```
java -classpath spy_dir\spy.jar com.ddtek.jdbcspy.SpyDriver
```

On Linux/UNIX:

```
java -classpath spy_dir/spy.jar com.ddtek.jdbcspy.SpyDriver
```

where *spy_dir* is the directory containing DataDirect Spy.

8 Developing JDBC Applications

This chapter provides information about developing JDBC applications for SequeLink environments including:

- ["JDBC 3.0 Support" on page 328](#)
- ["SQL Support" on page 329](#)
- ["Data Types and Isolation Levels" on page 331](#)
- ["Threading" on page 331](#)
- ["Using Scrollable Cursors" on page 333](#)
- ["Specifying Application IDs" on page 336](#)
- ["Error Handling" on page 342](#)
- ["Fine-Tuning JDBC Application Performance" on page 344](#)

JDBC 3.0 Support

The SequeLink *for* JDBC driver supports the JDBC 3.0 API. This functionality is available only to applications running on JDK 1.4.2 or higher Virtual Machines.

[Table 8-1](#) summarizes the JDBC 3.0 and enhanced JDBC 2.0 functionality that the JDBC driver supports.

Table 8-1. Supported JDBC Features

	DB2 UDB on z/OS	DB2 UDB Linux/UNIX /Windows	Informix	Oracle	Microsoft SQL Server	Sybase
Blob interfaces	X	X	X	X		
Clob interfaces	X	X	X	X		
DatabaseMetaData getSchema			X		X	X
setQueryTimeout		¹	X	X	X	X
ParameterMetaData	X	X	X		X	X
Savepoints	X	²		X	X	X
Updatable result sets	X	X	X	X	X	X

- 1. setQueryTimeout is supported on DB2 UDB only on Windows.
- 2. Only one savepoint can be released.

See [Appendix C “JDBC Support” on page 489](#) for information on the methods that SequeLink supports and the compatibility between the JDBC application versions.

JCA Resource Adapter Class

The `ManagedConnectionFactory` class for the SequeLink resource adapter is:

```
com.ddtek.resource.sljdbc.SequeLinkManagedConnectionFactory
```

See [“J2EE Connector Architecture Resource Adapter” on page 232](#) for information about using the JDBC Client as a JCA resource adapter.

SQL Support

See [Appendix A “SQL Escape Sequences” on page 431](#) for information about the SQL escape sequences supported by the JDBC driver.

Binding SQL Statements

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

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

The JDBC driver relies on the binding of parameters to know how to send information to the database system in its native format. The host variable data type and SQL data type are assumed to be the same. For example, `setInt()` indicates to the driver that the data type of the host variable and the SQL type of the dynamic parameter are both Integer. If the host variable and SQL data type are not the same, the application should use `setObject()` to specify the application variable data type and the SQL data type. If an application furnishes incorrect parameter binding information to the JDBC driver, the results will be unpredictable.

To ensure interoperability, the JDBC driver uses only the parameter binding information provided by the application. Although some DBMSs can publish dynamic parameter information back to a JDBC driver, others cannot. For example, both the SQL Server and Oracle query processors can determine that a parameter is an integer. However, the Oracle query processor cannot publish this information back to the *SequeLink* for JDBC driver.

NOTES:

- The SQL data type is determined at prepare time by the database and does not change for the life of the statement. The SQL data type is not dependent on the data being used by the application. For example, it is not valid to bind the SQL type to `SQL_NUMERIC` with a precision of 15 and a scale of 5, and then bind it on a later execution to a SQL type of `SQL_NUMERIC` with a precision of 13 and a scale of 3.
- You can implement `ParameterMetaData` only when a database system publishes parameter information after prepare time. The JDBC driver returns this information when the application requests it, but depending on the database, performance penalties can be incurred. You can tune this feature through the *SequeLink* data source service attribute `DataSourceDescribeParam`. Refer to the *SequeLink Administrator's Guide* for information on service attributes.

Data Types and Isolation Levels

The data types and isolation levels supported by the JDBC driver depend on the data store to which you are connecting. See [Appendix B “Data Types and Isolation Levels” on page 451](#) for database-specific information about data types and isolation levels.

Threading

The JDBC driver is completely thread safe; that is, it will not fail when database requests are made on separate threads.

Threading Architecture

A JDBC driver can be based on one of the following architectures:

- *Thread impaired.* The JDBC driver serializes all JDBC calls. All requests are handled one by one, without concurrent processing.
- *Thread per connection.* The JDBC driver processes requests concurrently with statements that do not share the same connection; however requests on the same connection are serialized. The SequeLink *for* JDBC driver uses this architecture.
- *Fully threaded.* All requests use the threaded model. The JDBC driver processes all requests on multiple statements concurrently.

Cancelling Functions in Multithreaded Applications

In a multithreaded application, a thread can use the cancel method to cancel a statement that is being executed by another thread. Whether the cancel method actually cancels the statement depends on the data store being accessed as shown in [Table 8-2](#).

- *OK* means that cancel can interrupt the running statement.
- *Ignored* means that cancel will have no affect on the running statement.

In both cases, the cancel method returns `SQL_SUCCESS`. If the cancel method has been called from a different thread while a request is pending, the original statement will return `SQL_ERROR` with the error message `Operation cancelled`.

Table 8-2. Using Cancel in Multithreaded JDBC Applications

Data Store	SQLCancel
DB2 UDB on z/OS	Ignored
DB2 UDB on Windows	OK
DB2 UDB on Linux/UNIX	Ignored
Informix	OK
Microsoft SQL Server	OK
Oracle on Windows	Ignored
Oracle on UNIX	OK
Sybase	OK

NOTE: Cancel functionality is not supported when the connection uses Secure Socket Layer (SSL) encryption.

Using Scrollable Cursors

Scrollable cursors can move backward and forward in a result set, allowing the application to scroll back and forth through retrieved data.

Result Set Types

JDBC defines the following result set types:

- Forward-only
- Scroll-insensitive
- Scroll-sensitive

Forward-only result sets allow you to move forward, but not backward, through the data. The application can move only forward using the `next()` method.

Typically, a *scroll-insensitive result set* ignores changes that are made while it is open. It provides a static view of the underlying data it contains. The membership, order, and column values of rows are fixed when the result set is created.

In contrast, a *scroll-sensitive result set* provides a dynamic view of the underlying data, reflecting changes that are made while it is open. The membership and ordering of rows in the result set may be fixed, depending on how it is implemented.

Support for the scroll-sensitive cursors depends on the data store to which you are connecting, as described in [Table 8-3](#).

Table 8-3. Support for Scroll-Sensitive Cursors (JDBC)

Database	Conditions for Support
DB2 UDB	The DB2 tables must contain an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.
Informix	None (inherently supported).
Microsoft SQL Server	The table must contain an identity column.
JDBC Socket	The backend database must support an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.
ODBC Socket	The backend database must support an auto-unique column. The name and other properties of the auto-unique column must be configured in the data source of the SequeLink Server.
Oracle	None (inherently supported).
Sybase	The table must contain an identity column.

Concurrency Types

JDBC defines the following concurrency types for a result set:

- Read-only
- Updatable

A *read-only* result set does not allow its contents to be updated. Read-only result sets can increase the overall level of concurrency between transactions, because multiple read-only locks can be held on a data item simultaneously.

An *updatable* result set allows its contents to be updated and may use database write locks to mediate access to the same data item by different transactions. Because only a single write lock may be held at one time on a data item, updatable result sets can reduce concurrency.

An optimistic concurrency control scheme may be appropriate if you can accurately predict that conflicting access to data will seldom occur. Typically, optimistic concurrency control implementations compare rows by a value or by a version number to determine if an update conflict has occurred.

Using Scrollable Cursors

- The JDBC driver supports forward-only and scroll-insensitive result sets against all data stores.
- Scroll-sensitive result sets on stored procedures or explicit batches are not supported.
- Scroll-sensitive result sets are not supported when the Select statement contains any of the following SQL language constructions:
 - JOIN
 - Aggregate functions
 - GROUP BY
- The JDBC driver supports updatable result sets.

NOTE: When the JDBC driver cannot support the requested result set type or concurrency, it will automatically downgrade it and generate one or multiple SQLWarnings with detailed information.

Specifying Application IDs

Application IDs are alphanumeric strings passed by a SequeLink Client that identify the client application to a SequeLink service that has been configured to accept connections only from specific application IDs.

After establishing a connection with the JDBC driver, immediately invoke `setApplicationId`. The `setApplicationId` method is defined on the interface `com.ddtek.jdbc.extensions.SlExtensionInterface`, and uses the following method prototype:

```
public void setApplicationId(String s) throws SQLException
```

You can set the application ID as shown in the following example:

```
import java.sql.*;
import com.ddtek.jdbc.extensions.SlExtensionInterface;

...
Connection con = DriverManager.getConnection(...);

String appId = "myAppID";
if (con instanceof SlExtensionInterface)
{
    SlExtensionInterface slCon = (SlExtensionInterface)con;
    slCon.setApplicationId(myAppID);
}
```

where *myAppID* is the application ID.

For more information about configuring SequeLink services to accept connections only from specific application IDs, refer to the *SequeLink Administrator's Guide*.

Parameter Metadata Support

The JDBC driver provides parameter metadata support for databases that provide native support for parameter metadata. For databases that do not provide native parameter metadata support, such as Oracle or Informix, the JDBC driver supports parameter metadata for valid ANSI SQL SELECT statements.

INSERT and UPDATE Statements

The JDBC driver supports returning parameter metadata for all types of SQL statements for databases that provide native support for parameter metadata.

For databases that do not provide native parameter metadata support, the JDBC driver supports returning parameter metadata for the following forms of Insert and Update statements:

- `INSERT INTO foo VALUES (?, ?, ?)`
- `INSERT INTO foo (col1, col2, col3) VALUES (?, ?, ?)`
- `UPDATE foo SET col1=?, col2=?, col3=? WHERE col1
operator ? [{AND | OR} col2 operator ?]`

where *operator* is any of the following SQL operators: =, <, >, <=, >=, and <>.

Select Statements

The JDBC driver supports returning parameter metadata for Select statements that contain parameters in ANSI SQL 92 entry-level predicates, for example, such as COMPARISON, BETWEEN, IN, LIKE, and EXISTS predicate constructs. Refer to the ANSI SQL reference for detailed syntax.

Parameter metadata can be returned for a Select statement if one of the following conditions is true:

- The statement contains a predicate value expression that can be targeted against the source tables in the associated FROM clause. For example:

```
SELECT * FROM foo WHERE bar > ?
```

In this case, the value expression "bar" can be targeted against the table "foo" to determine the appropriate metadata for the parameter.

- The statement contains a predicate value expression part that is a nested query. The nested query's metadata must describe a single column. For example:

```
SELECT * FROM foo WHERE (SELECT x FROM y
WHERE z = 1) < ?
```

The following Select statements show further examples for which parameter metadata can be returned:

```
SELECT col1, col2 FROM foo WHERE col1 = ? and col2 > ?
```

```
SELECT ... WHERE colname = (SELECT col2 FROM t2
WHERE col3 = ?)
```

```
SELECT ... WHERE colname LIKE ?
```

```
SELECT ... WHERE colname BETWEEN ? and ?
```

```
SELECT ... WHERE colname IN (?, ?, ?)
```

```
SELECT ... WHERE EXISTS(SELECT ... FROM T2 WHERE col1 < ?)
```

ANSI SQL 92 entry-level predicates in a WHERE clause containing GROUP BY, HAVING, or ORDER BY statements are supported. For example:

```
SELECT * FROM t1 WHERE col = ? ORDER BY 1
```

Joins are supported. For example:

```
SELECT * FROM t1,t2 WHERE t1.col1 = ?
```

Fully qualified names and aliases are supported. For example:

```
SELECT a, b, c, d FROM T1 AS A, T2 AS B WHERE A.a = ?  
and B.b = ?"
```

ResultSet Metadata Support

If your application requires table name information, the JDBC driver can return table name information in ResultSet metadata for Select statements. By setting the `ResultSetMetaDataOptions` property to 1, the JDBC driver performs additional processing to determine the correct table name for each column in the result set when the `ResultSetMetaData.getTableName()` method is called. Otherwise, the `getTableName()` method may return an empty string for each column in the result set.

The table name information that is returned by the JDBC driver depends on whether the column in a result set maps to a column in a table in the database. For each column in a result set that maps to a column in a table in the database, the JDBC driver returns the table name associated with that column. For columns in a result set that do not map to a column in a table (for example, aggregates and literals), the JDBC driver returns an empty string.

The Select statements for which ResultSet metadata is returned may contain aliases, joins, and fully qualified names. The following queries are examples of Select statements for which

the `ResultSetMetaData.getTableName()` method returns the correct table name for columns in the Select list:

```
SELECT id, name FROM Employee

SELECT E.id, E.name FROM Employee E

SELECT E.id, E.name AS EmployeeName FROM Employee E

SELECT E.id, E.name, I.location, I.phone FROM Employee E,
      EmployeeInfo I WHERE E.id = I.id

SELECT id, name, location, phone FROM Employee,
      EmployeeInfo WHERE id = empId

SELECT Employee.id, Employee.name, EmployeeInfo.location,
      EmployeeInfo.phone FROM Employee, EmployeeInfo .
WHERE Employee.id = EmployeeInfo.id
```

The table name returned by the driver for generated columns is an empty string. The following query is an example of a Select statement that returns a result set that contains a generated column (the column named "upper").

```
SELECT E.id, E.name as EmployeeName, {fn UCASE(E.name)}
      AS upper FROM Employee E
```

The JDBC driver also can return schema name and catalog name information when the `ResultSetMetaData.getSchemaName()` and `ResultSetMetaData.getCatalogName()` methods are called if the driver can determine that information. For example, for the following statement, the JDBC driver returns "test" for the catalog name, "test1" for the schema name, and "foo" for the table name:

```
SELECT * FROM test.test1.foo
```

The additional processing required to return table name, schema name, and catalog name information is only performed if the `ResultSetMetaData.getTableName()`, `ResultSetMetaData.getSchemaName()`, or `ResultSetMetaData.getCatalogName()` methods are called.

Unicode Support

Multi-lingual applications can be developed on any operating system platform with JDBC using the SequeLink *for* JDBC driver to access both Unicode and non-Unicode enabled databases. Internally, Java applications use UTF-16 Unicode encoding for string data. When fetching data, the JDBC driver automatically performs the conversion from the character encoding used by the database to UTF-16.

Similarly, when inserting or updating data in the database, the drivers automatically convert UTF-16 encoding to the character encoding used by the database.

The JDBC API provides mechanisms for retrieving and storing character data encoded as Unicode (UTF-16) or ASCII. Additionally, the Java string object contains methods for converting UTF-16 encoding of string data to or from many popular character encodings.

For information on configuring your server for full Unicode support, refer to the *SequeLink Administrator's Guide*.

Rowset Support

The JDBC driver supports any JSR 114 implementation of the RowSet interface, including:

- CachedRowSets
- FilteredRowSets
- WebRowSets
- JoinRowSets
- JDBCRowSets

J2SE 1.4 or higher is required to use rowsets with the driver.

See <http://www.jcp.org/en/jsr/detail?id=114> for more information about JSR 114.

Error Handling

The JDBC driver reports errors to the calling application by returning `SQLExceptions`. Errors can be generated by the following components:

- SequeLink *for* JDBC driver
- SequeLink Server
- Database

Driver Errors

An error generated by the JDBC driver has the following format:

```
[DataDirect] [SequeLink JDBC Driver] message
```

For example:

```
[DataDirect] [SequeLink JDBC Driver] Timeout expired.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*. Sometimes, you may need to check the last JDBC call your application made and refer to the JDBC specification for recommended action.

SequeLink® Server Errors

An error generated by SequeLink Server has the following format:

```
[DataDirect] [SequeLink JDBC Driver] [SequeLink Server]  
message
```

For example:

```
[DataDirect] [SequeLink JDBC Driver] [SequeLink Server]  
Only Select statements are allowed in this read-only  
connection.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

Database Errors

An error generated by the database has the following format:

```
[DataDirect] [SequeLink JDBC Driver] [...] message
```

For example:

```
[DataDirect] [SequeLink JDBC Driver] [Oracle]  
ORA-00942:table or view does not exist.
```

Use the native error code to look up details about the possible cause of the error. For these details, refer to your database documentation.

Fine-Tuning JDBC Application Performance

This section provides some tips for fine-tuning the performance of your JDBC applications.

Reducing Download Time

Generally, the time that it takes for applets to download is determined by the following factors:

- *Number of classes that are loaded.* Each class that is downloaded results in an HTTP request to your Web server. The more requests and transfers that are made, the slower the download.
- *Size of the byte code that is loaded.* The more bytes that are transferred, the slower the download.

JDK 1.2-compatible and higher Java Virtual Machines support JAR files, which reduces the number of HTTP requests because all the class files are packaged together in the JAR file. The JAR format also allows you to compress the packaged files, which further optimizes the download.

To reduce download time by using JAR files:

- 1 Package all classes of your applet into a JAR file.
- 2 Copy the JAR file into the directory indicated by the codebase tag.
- 3 Specify the JAR file in the archive tag.

For example:

```
<html>
<applet
width=100 height=100
code=MyApplet
codebase=.
archive=myapplet.jar>
<param name=ConfigFile value=Config.txt>
</applet>
```

The *SequeLink for JDBC* driver is packaged into the following JAR files:

- *sljc.jar* contains all classes of the *SequeLink for JDBC* driver.
- *slssl.jar* contains all classes of the *SequeLink for JDBC* driver implementation of SSL. This file is only required if you will be using SSL encryption.

To use the JDBC driver from within your applet, specify these JAR files in the archive tag as shown:

```
<html>
<applet
width=100 height=100
code=MyApplet
codebase=.
archive=myapplet.jar,sljc.jar,slssl.jar>
<param name=ConfigFile value=Config.txt>
</applet>
```

Fetching BigDecimal Objects

JDBC 1.22 defines `getBigDecimal()` with a scale parameter. When the JDBC driver fetches a `BigDecimal` object from a database, it rescales it using the scale specified by the application. This additional processing can downgrade system performance, particularly when large numbers of `BigDecimal` objects are fetched by your application.

To eliminate this additional rescaling, JDBC 2.0 defines an overloaded version of `getBigDecimal`, without the scale parameter. This method allows the JDBC driver to return the `BigDecimal` object with the original precision.

Using Database Metadata Methods

Because database metadata methods that generate `ResultSet` objects are slow compared to other JDBC methods, their frequent use can impair system performance. The guidelines in this section will help you to optimize system performance when selecting and using database metadata.

Minimizing the Use of Database Metadata Methods

Compared to other JDBC methods, database metadata methods that generate `ResultSet` objects are relatively slow. Applications should cache information returned from result sets that generate database metadata methods so that multiple executions are not needed.

While almost no JDBC application can be written without database metadata methods, you can improve system performance by minimizing their use. To return all result column information *mandated* by the JDBC specification, a JDBC driver may have to perform complex queries or multiple queries to

return the necessary result set for a single call to a database metadata method. These particular elements of the SQL language are performance expensive.

Applications should cache information from database metadata methods. For example, call `getTypeInfo` 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 database metadata method, so the cache of information should not be difficult to maintain.

Avoiding Search Patterns

Using null arguments or search patterns in database metadata methods results in generating time-consuming queries. In addition, network traffic potentially increases due to unwanted results. Always supply as many non-null arguments to result sets that generate database metadata methods as possible.

Because database metadata methods 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:

```
ResultSet WSrs = WSc.getTables (null, null, "WSTable", null);
```

should be:

```
ResultSet WSrs = WSc.getTables ("cat1", "johng", "WSTable", "TABLE");
```

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 database metadata methods can result in improved performance and reliability.

Using a Dummy Query to Determine Table Characteristics

Avoid using `getColumns` to determine characteristics about a table. Instead, use a dummy query with `getMetadata`.

Consider an application that allows the user to choose columns. Should the application use `getColumns` to return information about the columns to the user or instead prepare a dummy query and call `getMetadata`?

Case 1: GetColumns Method

```
ResultSet WSrc = WSc.getColumns (... "UnknownTable" ...);
// This call to getColumns will generate a query to
// the system catalogs... possibly a join
// which must be prepared, executed, and produce
// a result set
. . .
WSrc.next();
string Cname = getString(4);
. . .
// user must retrieve N rows from the server
// N = # result columns of UnknownTable
// result column information has now been obtained
```

Case 2: GetMetadata Method

```
// prepare dummy query
PreparedStatement WSps = WSc.prepareStatement
    (... "SELECT * from UnknownTable WHERE 1 = 0" ...);
// query is never executed on the server -
// only prepared
ResultSetMetaData WSsmd=wsps.getMetaData();
int numcols = WSsmd.getColumnCount();
...
int ctype = WSsmd.getColumnType(n)
...
```

```
// result column information has now been obtained
// Note we also know the column ordering within the table!
// This information cannot be assumed from the getColumns 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 this situation, Case 2 still outperforms Case 1.

Retrieving Data

This section provides general guidelines for retrieving data with JDBC applications.

Retrieving Long Data

Unless it is necessary, applications should not request long data because retrieving long data across a network is slow and resource-intensive.

Most users don't want to see long data. If the user does need to see these result items, the application can query the database again, specifying only the long columns in the select list. This method allows the average user to retrieve result sets 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 JDBC driver (for example, some

applications `SELECT * FROM table name ...`). If the select list contains long data, the driver must retrieve that data at fetch time, even if the application does not get the long data in the result set. When possible, the application developer should use a method that does not retrieve all columns of the table.

Additionally, although the `getClob` and `getBlob` methods allow the application to control how long data is retrieved in the application, the designer must realize that in many cases, the JDBC driver emulates these methods due to the lack of true locator support in the DBMS. In such cases, the driver must retrieve all of the long data across the network before exposing the `getClob` and `getBlob` methods.

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.

Reducing the Size of Data Retrieved

To reduce network traffic and improve performance, you can reduce the size of data being retrieved to a manageable limit by calling `setMaxRows`, `setMaxFieldSize`, and the driver-specific `SetFetchSize`. Another method of reducing the size of the data being retrieved is to decrease the column size. If the driver allows you to define the packet size, use the smallest packet size that will meet your needs.

In addition, be careful to return only the rows you need. If you return five columns when you only need two columns, performance is decreased, especially if the unnecessary rows include long data.

Choosing the Right Data Type

Retrieving and sending certain data types can be expensive. When you design a schema, select the data type that can be processed most efficiently. For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

Processing time is shortest for character strings, followed by integers, which usually require some conversion or byte ordering. Processing floating-point data and timestamps is at least twice as slow as integers.

Selecting JDBC Objects and Methods

The guidelines in this section will help you to optimize system performance when selecting and using JDBC objects and methods.

Using Parameter Markers as Arguments to Stored Procedures

When calling stored procedures, always use parameter markers for the argument markers instead of using literal arguments. JDBC drivers can call stored procedures on the database server either by executing the procedure as any other SQL query, or by optimizing the execution by invoking a Remote Procedure Call (RPC) directly into the database server. Executing the stored procedure as a SQL query results in the database server parsing the statement, validating the argument types, and converting the arguments into the correct data types. Remember that SQL is always sent to the database server as a character string, for example, "{call getCustName (12345)}". In this case, even

though the application programmer might assume that the only argument to `getCustName` is an integer, the argument is actually passed inside a character string to the server. The database server would parse the SQL query, isolate the single argument value `12345`, then convert the string `'12345'` into an integer value.

By invoking an RPC inside the database server, the overhead of using a SQL character string is avoided. Instead, the procedure is called only by name with the argument values already encoded into their native data types.

Case 1

Stored Procedure cannot be optimized to use a server-side RPC. The database server must parse the statement, validate the argument types, and convert the arguments into the correct data types. The database server must parse the statement, validate the argument types, and convert the arguments into the correct data types.

```
CallableStatement cstmt = conn.prepareCall ("call getCustName (12345)");
ResultSet rs = cstmt.executeQuery ();
```

Case 2

Stored Procedure can be optimized to use a server-side RPC. Because the application calls the procedure by name and the argument values are already encoded, the load on the database server is less.

```
CallableStatement cstmt = conn.prepareCall ("Call getCustName (?)");
cstmt.setLong (1,12345);
ResultSet rs = cstmt.executeQuery();
```


Using the Statement Object instead of the PreparedStatement Object

JDBC drivers are optimized based on the perceived use of the functions that are being executed. Choose between the `PreparedStatement` object and the `Statement` object depending on the planned use. The `Statement` object is optimized for a single execution of a SQL statement. In contrast, the `PreparedStatement` object is optimized for SQL statements that will be executed two or more times.

The overhead for the initial execution of a `PreparedStatement` object is high. The advantage comes with subsequent executions of the SQL statement.

Choosing the Right Cursor

Choosing the appropriate type of cursor allows maximum application flexibility. This section summarizes the performance issues of three types of cursors.

A forward-only cursor provides excellent performance for sequential reads of all of the rows in a table. However, it cannot be used when the rows to be returned are not sequential.

Insensitive cursors used by JDBC drivers are ideal for applications that require high levels of concurrency on the database server and require the ability to scroll forwards and backwards through result sets. The first request to an insensitive cursor fetches all of the rows and stores them on the client. Thus, the first request is very slow, especially when long data is retrieved. Subsequent requests do not require any network traffic and are processed quickly. Because the first request is processed slowly, insensitive cursors should not be used for a single request of one row. Designers should also avoid using insensitive cursors when long data is returned, because memory can be exhausted. Some insensitive cursor implementations cache the data in a

temporary table on the database server and avoid the performance issue.

Sensitive cursors, sometimes called keyset-driven cursors, use identifiers, such as a ROWID, that already exist in your database. When you scroll through the result set, the data for the identifiers is retrieved. Because each request generates network traffic, performance can be very slow. However, returning nonsequential rows does not further affect performance. Sensitive cursors are the preferred scrollable cursor model for dynamic situations, when the application cannot afford to buffer the data from an insensitive cursor.

Using get Methods Effectively

JDBC provides a variety of methods to retrieve data from a result set, such as `getInt`, `getString`, and `getObject`. The `getObject` method is the most generic and provides the worst performance when the non-default mappings are specified. This is because the JDBC driver must do extra processing to determine the type of the value being retrieved and generate the appropriate mapping. Always use the specific method for the data type.

To further improve performance, provide the column number of the column being retrieved, for example, `getString(1)`, `getLong(2)`, and `getInt(3)`, instead of the column name. If the column names are not specified, network traffic is unaffected, but costly conversions and lookups increase. For example, suppose you use:

```
getString("foo")...
```

The JDBC driver may have to convert `foo` to uppercase (if necessary), and then compare `foo` with all the columns in the column list. That is costly. If, instead, the driver was able to go directly to result column 23, then a lot of processing would be saved.

For example, suppose you have a result set that has 15 columns and 100 rows, and the column names are not included in the result set. You are interested in three columns, EMPLOYEE_NAME (a string), EMPLOYEE_NUMBER (a long integer), and SALARY (an integer). If you specify `getString("EmployeeName")`, `getLong("EmployeeNumber")`, and `getInt("Salary")`, each column name must be converted to uppercase, and lookups would increase considerably. Performance would improve significantly if you specify `getString(1)`, `getLong(2)`, and `getInt(15)`.

Designing JDBC Applications

The guidelines in this section will help you to optimize system performance when designing JDBC applications.

Managing Connections

Connection management is important to application performance. Optimize your application by connecting once and using multiple statement objects, instead of performing multiple connections. Avoid connecting to a data source after establishing an initial connection.

Although gathering driver information at connect time is a good practice, it is often more efficient to gather it in one step rather than two steps. For example, some applications establish a connection and then call a method in a separate component that reattaches and gathers information about the driver. Applications that are designed as separate entities should pass the established connection object to the data collection routine instead of establishing a second connection.

Another bad practice is to connect and disconnect several times throughout your application to perform SQL statements. Connection objects can have multiple statement objects

associated with them. Statement objects, which are defined to be memory storage for information about SQL statements, can manage multiple SQL statements.

You can improve performance significantly with connection pooling, especially for applications that connect over a network or through the World Wide Web. Connection pooling lets you reuse connections. Closing connections does not close the physical connection to the database. When an application requests a connection, an active connection is reused, thus avoiding the network input/output needed to create a new connection.

Connection and statement handling should be addressed before implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

Managing Commits in Transactions

Committing transactions is extremely disk I/O intensive and slow. Always turn off autocommit by using the following setting:

```
WSConnection.setAutoCommit(false).
```

What does a commit actually involve? The database server must flush back to disk every data page that contains updated or new data. This is not a sequential write but a searched write to replace existing data in the table. By default, Autocommit is on when connecting to a data source, and Autocommit mode usually impairs performance because of the significant amount of disk input/output needed to commit every operation.

Furthermore, some database servers do not provide an Autocommit mode. For this type of server, the JDBC driver must explicitly issue a COMMIT statement and a BEGIN TRANSACTION statement for every operation sent to the server. In addition to the large amount of disk input/output required to support

Autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

Although using transactions can help application performance, do not take this tip too far. Leaving transactions active can reduce throughput by holding locks on rows for long times, preventing other users from accessing the rows. Commit transactions in intervals that allow maximum concurrency.

Choosing the Right Transaction Model

Many systems support distributed transactions; that is, transactions that span multiple connections. Distributed transactions are at least four times slower than normal transactions due to the logging and network input/output necessary to communicate between all the components involved in the distributed transaction. Unless distributed transactions are required, avoid using them. Instead, use local transactions whenever possible.

For the best system performance, design the application to run under a single Connection object.

Updating Data

This section provides general guidelines to help you to optimize system performance when updating data in databases.

Using updateXXX Methods

Although programmatic updates do not apply to all types of applications, developers should attempt to use programmatic updates and deletes. Using the updateXXX methods of the ResultSet object allows the developer to update data without building a complex SQL statement. Instead, the developer simply

supplies the column in the result set that is to be updated and the data that is to be changed. Then, before moving the cursor from the row in the result set, the `updateRow` method must be called to update the database as well.

In the following code fragment, the value of the `Age` column of the `ResultSet` object `rs` is retrieved using the method `getInt`, and the method `updateInt` is used to update the column with an `int` value of 25. The method `updateRow` is called to update the row in the database that contains the modified value.

```
int n = rs.getInt("Age");
// n contains value of Age column in the resultset rs
. . .
rs.updateInt("Age", 25);
rs.updateRow();
```

In addition to making the application more easily maintainable, programmatic updates usually result in improved performance. Because the database server is already positioned on the row for the `Select` statement in process, performance-expensive operations to locate the row to be changed are not needed. If the row must be located, the server usually has an internal pointer to the row available (for example, `ROWID`).

Using `getBestRowIdentifier()`

Use `getBestRowIdentifier()` to determine the optimal set of columns to use in the `Where` clause for updating data. Pseudo-columns often provide the fastest access to the data, and these columns can only be determined by using `getBestRowIdentifier()`.

Some applications cannot be designed to take advantage of positional updates and deletes. Some applications might formulate the `Where` clause by using all searchable result columns by calling `getPrimaryKeys()`, or by calling `getIndexInfo()` to find columns that might be part of a unique index. These methods usually work, but might result in fairly complex queries.

Consider the following example:

```
ResultSet WSrs = WSs.executeQuery
    ("SELECT first_name, last_name, ssn, address, city, state, zip
     FROM emp");
// fetchdata
...
WSs.executeUpdate ("UPDATE EMP SET ADDRESS = ?
    WHERE first_name = ? and last_name = ? and ssn = ?
    and address = ? and city = ? and state = ?
    and zip = ?");
// fairly complex query
```

Applications should call `getBestRowIdentifier()` to retrieve the optimal set of columns (possibly a pseudo-column) that identifies a specific 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 generally 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 `getColumns`. To determine if pseudo-columns exist, call `getBestRowIdentifier()`.

Consider the previous example again:

```
...
ResultSet WSrowid = getBestRowIdentifier()
    (.... "emp", ...);
// Suppose this returned "ROWID"
...
ResultSet WSrs = WSs.executeQuery("SELECT first_name, last_name,
    ssn, address, city, state, zip, ROWID FROM emp");
// fetch data and probably "hide" ROWID from the user
...
WSs.executeUpdate ("UPDATE emp SET address = ? WHERE ROWID = ?");
// fastest access to the data!
```

If your data source does not contain special pseudo-columns, then the result set of `getBestRowIdentifier()` consists of the columns of the most optimal unique index on the specified table (if a unique index exists). Therefore, your application does not need to call `getIndexInfo` to find the smallest unique index.

Part 4: Developing .NET Applications

This part contains the following chapters:

- [Chapter 9 “Using the .NET Client” on page 363](#) provides information about using .NET applications with the SequeLink Client *for* .NET.
- [Chapter 10 “Developing .NET Applications” on page 389](#) provides information about developing .NET applications for SequeLink environments.

9 Using the .NET Client

This chapter provides information on using the SequeLink Client for .NET (the .NET Client).

About the .NET Client

The .NET Client provides .NET access through any .NET-enabled application or application server. It delivers high-performance point-to-point and *n*-tier access to industry-leading data stores across the Internet and intranets. The .NET Client is optimized for the .NET environment, allowing you to incorporate .NET technology and extend the functionality and performance of your existing system. The .NET Client also supports tracing.

The machine must have the .NET Framework Version 1.0 or 1.1 installed.

The .NET Client is compliant with the Microsoft .NET Framework Version 1.0 and 1.1 Software Development Kit (SDK).

Using Connection Pooling

Connection pooling allows you to reuse connections rather than create a new one every time the data provider needs to establish a connection to the underlying database. The data provider automatically enables connection pooling for your .NET client application.

You can control connection pooling behavior by using connection string options (see [“Specifying Connection Options” on page 369](#)). For example, you can define the number of connection pools, the number of connections in a pool, and the lifetime of pooled connections.

Connection pooling in ADO.NET is not provided by the .NET Framework. It must be implemented in the .NET data provider itself.

Creating a Connection Pool

Each connection pool is associated with a specific connection string. By default, the connection pool is created when the first connection with a unique connection string connects to the database. The pool is populated with connections up to the minimum pool size. Additional connections can be added until the pool reaches the maximum pool size.

The pool remains active as long as any connections remain open, either in the pool or used by an application with a reference to a Connection object that has an open connection.

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

In the following C# code fragment, three new `SequeLinkConnection` objects are created, but only two connection pools are required to manage them. Note that the first and second connection strings differ only by the value assigned for User ID and Password, and by the value of the Min Pool Size option.

```
SequeLinkConnection conn1 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                        User ID=john;password=beach;
                        Database=test;Min Pool Size=50";

conn1.Open();
// Pool A is created and filled with connections to
// the minimum pool size.

SequeLinkConnection conn2 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                        User ID=mary;password=jtb28cnc
                        Database=test;Min Pool Size=100";

conn2.Open();
// Pool B is created because the connection strings differ.

SequeLinkConnection conn3 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                        User ID=john;password=beach;
                        Database=test;Min Pool Size=50";

conn3.Open();
// Conn3 is assigned an existing connection that was
// created in Pool A when the pool was created for Conn1
```

Adding Connections to a Pool

A connection pool is created in the process of creating each unique connection string that an application uses. When a pool is created, it is populated with enough connections to satisfy the minimum pool size requirement, set by the Min Pool Size connection string option. If an application is using more connections than Min Pool Size, the data provider allocates

additional connections to the pool up to the value of the Max Pool Size connection string option, which sets the maximum number of connections in the pool.

When a `SequeLinkConnection` object is requested by the application calling the `Connection.Open(...)` method, the connection is obtained from the pool, if a usable connection is available. A usable connection is defined as a connection that is not currently in use by another valid `SequeLinkConnection` object, has a matching distributed transaction context (if applicable), and has a valid link to the server.

If the maximum pool size has been reached and no usable connection is available, the request is queued in the data provider. The data provider waits for the value of the `Connection Timeout` connection string option for a usable connection to return to the application. If this time period expires and no connection has become available, then the data provider returns an error to the application.

IMPORTANT: Closing the connection using the `Close()` or `Dispose()` method of the `Connection` object adds or returns the connection to the pool. When the application uses the `Close` method, the connection string settings remain as they were before the `Open` was called. If you use the `Dispose` method to close the connection, the connection string settings are cleared, and the default settings are restored.

Removing Connections from a Pool

A connection is removed from a connection pool when it either exceeds its lifetime as determined by the `Connection Lifetime` connection string option, or when a new connection is initiated by the application (`SequeLinkConnection.Open()` is called) that has a matching connection string.

Before returning a connection from the connection pool to an application, the Pool Manager checks to see if the connection has

been closed at the server. If the connection is no longer valid, the Pool Manager discards it and returns another connection from the pool, if one is available and valid.

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

Handling Dead Connection in a Pool

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

The .NET data provider handles this situation transparently to the user. The application does not receive any errors on the Connection.Open() attempt because the data provider simply returns a connection from a connection pool. The first time the Connection object is used to execute a SQL statement (for example, through one of the DataReader execution methods or the DataAdapter.Fill method), the data provider detects that the physical connection to the server has been lost and attempts to reconnect to the server before executing the SQL statement. If the data provider can reconnect to the server, the result of the SQL execution is returned to the application; no errors are returned to the application. The data provider uses the connection failover options, if enabled, when attempting this seamless reconnection. See [“Using Connection Failover” on page 375](#) for information about configuring the data provider to connect to a backup server when the primary server is not available.

NOTE: Because the .NET Client can attempt to reconnect to the database server when executing SQL statements, connection errors can be returned to the application when a statement is executed. If the .NET Client cannot reconnect to the server (for example, because the server is still down), the execution method throws an error indicating that the reconnect attempt failed, along with specifics about the reason the connection failed.

DataDirect's method of handling dead connections in connection pools allows for the maximum performance out of the connection pooling mechanism. Some data providers periodically ping the server with a dummy SQL statement while the connections sit idle. Other data providers ping the server when the application requests the use of the connection from the connection pool. Both of these approaches add round trips to the database server and ultimately slow down the application during normal operation of the application is occurring.

Handling Distributed Transactions in a Pool

The Pool Manager groups the connections according to the requirement for transactions. If the requesting thread requires a specific transaction context, it must be matched to a connection with the same transaction context, for example, a connection that has been enlisted in distributed transactions.

Because closed connections are returned to the appropriate connection pool, you can close a connection even though a distributed transaction is pending. This means that you can still commit or roll back the distributed transaction until the connection is closed at the server.

See [“Using Distributed Transactions” on page 381](#) for more information about how the data provider processes distributed transactions

Tracking Connection Pool Performance

All DataDirect ADO.NET data providers install a set of PerfMon counters that let you tune and debug applications that use the data provider. See [“PerfMon Support” on page 410](#) for information about using the PerfMon counters.

Specifying Connection Options

You can modify a connection by specifying connection string options. See [“.NET Public Objects/Interfaces Supported” on page 396](#) for information about specifying options through the Client’s SequeLinkConnection object.

The basic format of a connection string includes a series of keyword/value pairs separated by semicolons. The following example shows the keywords and values for a simple connection string to connect to the host hal:

```
"Host=hal;Port=19998;User Id=test01;Password=test01;Database=test;"
```

Use the following guidelines when specifying a connection string:

- The spaces in the connection string option names are required.
- All connection string option names are case-insensitive. For example, Password is the same as password. However, the values of options such as User ID and Password may be case-sensitive.
- Special characters can be used in the value of the connection string option. To escape special characters, surround the value in double quotes.

Table 9-1 gives the names for each connection string option, as well as a description. The defaults listed are initial defaults that apply when no value is specified in the connection string.

Table 9-1. .NET Connection String Options	
Option	Description
Alternate Servers	<p>Specifies a list of alternate database servers to which the data provider will try to connect if the primary SequeLink server is unavailable. Specifying a value for this connection string option enables connection failover for the data provider.</p> <p>The value you specify must be in the form of a string that defines the physical location of each alternate server. You must specify the server name or the IP address and port number of each alternate SequeLink server. All of the other required connection information for each alternate server is the same as what is defined for the primary server connection. Currently, the only optional connection string option that can be set for the alternate server is Server Data Source. The string has the format:</p> <pre>"Host=servername1;Port=port1[;ServerDataSource=serverdsname]"</pre> <p>For example, the following connection string defines two alternate SequeLink servers for connection failover:</p> <pre>"Host=server1;port=19996;ServerDataSource=SDSN1;User ID=test;Password=secret;Alternate Servers=(Host=server2:Port=19996:ServerData Source=SDSN2,Host=server3:Port=19996:ServerDataSource=SDSN3) "</pre> <p>See “Using Connection Failover” on page 375 for a discussion of connection failover and for information about other connection string options that you can set for this feature.</p>
Connection Lifetime	<p>Specifies the number of seconds to keep connections in a connection pool. The pool manager periodically checks all pools, and closes and removes any connection that exceeds its lifetime. The Min Pool Size connection string option can cause some connections to ignore this value.</p> <p>Valid values are 0 to 65335.</p> <p>When set to 0 (the initial default), the lifetime is never limited by time.</p>

Table 9-1. .NET Connection String Options (cont.)

Option	Description
Connection Reset	<p>Connection Reset={True False}. Specifies whether a connection that is removed from the connection pool for reuse by an application will have its state reset to the initial configuration settings of the connection. Resetting the state impacts performance of the reused connections because the new connection must issue additional commands to the server.</p> <p>When set to false (the initial default), the data provider does not reset the state of the connection.</p>
Connection Retry Count	<p>Specifies the number of times the data provider tries to connect to the primary server, and, if specified, the alternate servers after the initial unsuccessful attempt.</p> <p>The value can be any integer from 0 to 65535.</p> <p>When set to 0 (the initial default), the data provider does not try to reconnect after the initial unsuccessful attempt.</p> <p>If a connection is not established during the retry attempts, the data provider returns an error that is generated by the last server to which it attempted to connect.</p> <p>This option and Connection Retry Delay, which specifies the wait interval between attempts, can be used in conjunction with connection failover. Refer to the <i>SequeLink Administrator's Guide</i> for a discussion of connection failover and for information about other connection string options that you can set for this feature</p>

Table 9-1. .NET Connection String Options (cont.)

Option	Description
Connection Retry Delay	<p>Specifies the number of seconds the data provider waits after the initial unsuccessful connection attempt before retrying a connection to the primary server, and, if specified, the alternate servers.</p> <p>The value can be any integer from 0 to 65535.</p> <p>The initial default is 3 (seconds). When set to 0, there is no delay between retrying the connection.</p> <p>NOTES:</p> <ul style="list-style-type: none">■ This option has no effect unless the Connection Retry Count connection string option is set to an integer value greater than 0.■ This option and the Connection Retry Count connection string option, which specifies the number of times the data provider attempts to connect after the initial attempt, can be used in conjunction with connection failover. Refer to the <i>SequeLink Administrator's Guide</i> for a discussion of connection failover and for information about other connection string options that you can set for this feature.
Database	<p>Specifies the name of the database to which you want to connect.</p>
Enlist	<p>Enlist={True False}. Specifies whether the data provider automatically attempts to enlist the connection in creating the thread's current transaction context. This connection string option is enabled only when the data provider is installed with a Server license and the optional MS DTC Support components were selected.</p> <p>When set to false (the initial default), the data provider does not automatically attempt to enlist the connection.</p> <p>NOTE: Enlisting in distributed transactions requires the data provider to call unmanaged code.</p>
Host	<p>Specifies either the IP address or the name of the server to which you want to connect. For example, if your network supports named servers, you can specify a server name such as <code>SequeLinkAppServer</code>. Or, you can specify an IP address such as <code>122.23.15.12</code>.</p> <p>The initial default is localhost, which specifies a SequeLink server running on the local machine.</p>

Table 9-1. .NET Connection String Options (cont.)

Option	Description
License Path	<p>Specifies the fully-qualified path to the DDTEK.LIC license file. The license file is installed by default in the product installation directory, for example, C:\install_dir\DDTek.lic.</p> <p>If you do not provide this option, the data provider looks for the license file in the application's current directory. If the license file is not found, the data provider checks for keys placed in the registry during the installation process; then, the data provider looks for the license key in the installation directory. If the license key is still not found, the data provider fails to connect.</p> <p>The initial default is an empty string.</p>
Load Balancing	<p>Load Balancing={true false}. Determines whether the data provider uses client load balancing in its attempts to connect to primary and alternate SequeLink servers. The list of alternate servers is specified by the Alternate Servers connection option.</p> <p>When set to true, the data provider attempts to connect to the database servers in random order.</p> <p>When set to false (the initial default), client load balancing is not used and the data provider connects to each server based on its sequential order (primary server first, then, alternate servers in the order they are specified).</p> <p>NOTE: This option has no effect unless alternate servers are defined for the Alternate Servers connection string option.</p> <p>The Load Balancing connection string option is an optional setting that you can use in conjunction with connection failover. Refer to the <i>SequeLink Administrator's Guide</i> for more information for a discussion of connection failover and for information about other connection options that you can set for this feature.</p>
Max Number Of Pools	<p>Specifies the maximum number of connection pools that can be in use at a time during the life of the process.</p> <p>The value can be any integer from 1 to 65335.</p> <p>The initial default is 100.</p> <p>See "Creating a Connection Pool" on page 364 for more information about creating connection pools.</p>

Table 9-1. .NET Connection String Options (cont.)

Option	Description
Max Pool Size	<p>Specifies the maximum number of connections within a single pool. When the maximum number is reached, no additional connections can be added to the connection pool.</p> <p>The value can be any integer from 1 to 65335.</p> <p>The initial default is 100.</p> <p>See “Creating a Connection Pool” on page 364 for more information about creating connection pools.</p>
Min Pool Size	<p>Specifies the minimum number of connections are opened and placed in a connection pool when it is created. The connection pool retains this number of connections, even when some connections exceed their Connection Lifetime value.</p> <p>The value can be any integer from 0 to 65335.</p> <p>When set to 0 (the initial default), no additional connections are placed in the connection pool when it is created.</p> <p>See “Removing Connections from a Pool” on page 366 for a discussion of connection lifetimes.</p>
MS Pooling	<p>MS Pooling={true false}. Specifies whether the data provider uses the legacy DataDirect connection pooling implementation (see “Using Connection Failover” on page 375).</p> <p>When set to false, the data provider uses the legacy connection pool model, which is provided only for backward compatibility.</p> <p>When set to true (the initial default), the data provider does not use the legacy connection pool model.</p>
Password	<p>Specifies the host or data store password, which may be required depending on the server configuration.</p>
Pooling	<p>Pooling={True False}. Specifies whether connections are pooled. Connection pooling can significantly enhance the performance and scalability of your application.</p> <p>When set to True (the initial default), connection pooling is enabled.</p>
Port	<p>Specifies the TCP/IP port on which the SequeLink Server is listening.</p> <p>The initial default is 19996.</p>

Table 9-1. .NET Connection String Options (cont.)

Option	Description
ServerDataSource	Specifies the server data source to be used by the connection. If not specified, the configuration of the default server data source are used.
User ID	Specifies the SequeLink User ID for this connection, which may be required depending on the server configuration.

Using Connection Failover

Connection failover allows an application to connect to an alternate, or backup, SequeLink Server if the primary SequeLink Server is unavailable, for example, because of a hardware failure or traffic overload.

Refer to the *SequeLink Administrator's Guide* for more information about connection failover.

To configure connection failover to another server, you must specify a list of alternate database servers that are tried at connection time if the primary server is not accepting connections. To do this, use the Alternate Servers connection string option. Connection attempts continue until a connection is successfully established or until all the databases in the list have been tried once (the default).

Optionally, you can specify the following additional connection failover features:

- The number of times the data provider attempts to connect after the initial connection attempt. By default, the data provider does not retry. To set this feature, use the Connection Retry Count connection string option.

- The wait interval, in seconds, used between attempts to connect. The default interval is 3 seconds. To set this feature, use the Connection Retry Delay connection string option.
- Whether the .NET Client will use load balancing in its attempts to connect to primary and alternate SequeLink Servers. If load balancing is enabled, the .NET Client uses a random pattern instead of a sequential pattern in its attempts to connect. The default value is not to use load balancing. To set this feature, use the Load Balancing connection string option.

You use a connection string to direct the .NET Client to use connection failover. See [“Specifying Connection Options” on page 369](#) for information about using connection strings.

The following C# code fragment includes a connection string that configures the .NET Client to use connection failover in conjunction with all of its optional features—load balancing, connection retry, and connection retry delay:

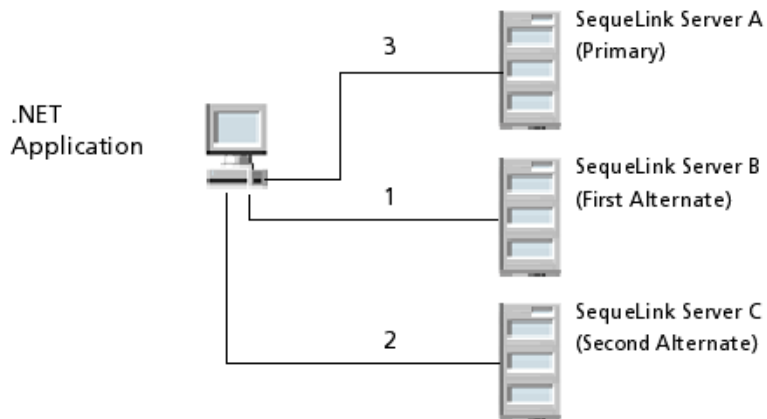
```
SequeLinkConnection Conn = new SequeLinkConnection();
Conn = new SequeLinkConnection("Host=server1;port=19996;ServerDataSource=
SDSN1;User ID=test;Password=secret;Alternate Servers=(Host=server2:Port=
19996:ServerDataSource=SDSN2,Host=server3:Port=19996:ServerDataSource=
SDSN3);Connection Retry Count=4;Connection Retry Delay=5;Load Balancing=true;
Connection Timeout=60")
```

Specifically, this connection string configures the .NET Client to use two alternate servers as connection failover servers, to attempt to connect four additional times if the initial attempt fails, to wait five seconds between attempts, and to try the primary and alternate servers in a random order. Each connection attempt lasts for 60 seconds.

Client Load Balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When both connection failover and client load balancing are enabled, the first connection attempt establishes the random order in which primary and alternate SequeLink Servers are tried. Subsequent connection attempts use the same sequence. For example, suppose that client load balancing is enabled as shown in [Figure 9-1](#).

Figure 9-1. Client Load Balancing Example



First, SequeLink Server B is tried (1). Then, SequeLink Server C may be tried (2), followed by a connection attempt to SequeLink Server A (3); subsequent connection attempts use this same sequence. In contrast, if client load balancing was not enabled in this scenario, each SequeLink Server would be tried in sequential order, primary server first, then alternate servers based on their entry order in the alternate servers list.

Using .NET Objects

The .NET Client supports the .NET public objects. See [“.NET Public Objects/Interfaces Supported” on page 396](#) for more information.

The .NET Client exposes its objects as sealed objects.

Assemblies

A .NET assembly is a compiled representation of one or more classes. Each assembly is self-contained, that is, the assembly includes the metadata about the assembly as a whole.

Assemblies can be private or shared. *Private assemblies*, which are used by a limited number of applications, are placed in the application folder or one of its subfolders. Even if the client has two different applications that call a private assembly named formulas, each client application loads the correct assembly.

Shared assemblies, which are available to multiple client applications, are placed in the Global Assembly Cache (GAC). Each shared assembly is assigned a strong name to handle name and version conflicts.

The assembly name for the .NET data provider is ddtek.sequelink.dll. The assembly, which is placed in the Global Assembly Cache (GAC) during installation, is strongly named.

Parameter Markers

Parameter markers, including parameter markers for stored procedures, are specified in the .NET data provider by using the "?" symbol in SQL statements.

```
UPDATE emp SET job = ?, sal = ? WHERE empno = ?
```

Because parameters are not named, the bindings must occur in the order of the parameters in the statement. This means that the calls to the Add() method on the ParameterCollection object (adding the Parameter objects to the collection) must occur in the order of the "?"s in the statement. The name passed to the Add() method does not need to match anything in the SQL statement.

Parameter Arrays

Parameter array binding is typically used with Insert statements to speed up the time needed to fill a table. An application can specify rows of parameter values with a single execution of a command. The values can then be sent to the database server in a single round trip (depending on the native capabilities of the backend database).

The .NET Client supports input parameter arrays for Insert and Update statements. The data provider supports output arrays for stored procedures.

Stored Procedures

NOTE: The .NET Client attaches the provider-specific prefix "SequeLink" to the names of the public objects, for example, SequeLinkConnection or SequeLinkCommand.

To enable stored procedures in the application, do the following:

- Set the `CommandText` property in the `SequeLinkCommand` object to the stored procedure name.
- Set the `CommandType` property in the `SequeLinkCommand` object to `StoredProcedure`.
- Specify parameter arguments, if needed. The application should Add the parameters to the parameter collection of the `SequeLinkCommand` object in the order of the arguments to the stored procedure. The application does not need to specify the parameter markers in the `CommandText` property of the `SequeLinkCommand` object.

To retrieve the return value from a stored procedure, the application should add an extra parameter to the parameter collection for the `SequeLinkCommand` object. This parameter's `ParameterDirection` property should be set to `ParameterDirection.ReturnValue`. The return value parameter can be anywhere in the parameter collection because it does not correspond to a specific parameter marker in the `Text` property of the `SequeLinkCommand` object.

If the stored procedure does not produce a return value, parameters bound with the `ParameterDirection.ReturnValue` property are ignored.

If the stored procedure returns a `ReturnValue` from the database and the application has not bound a parameter for it, the data provider discards the value.

Transaction Support

NOTE: The .NET Client attaches the provider-specific prefix "SequeLink" to the names of the public objects, for example, SequeLinkConnection or SequeLinkCommand.

The .NET Client supports both local and distributed transactions.

Using Local Transactions

The .NET Client uses 100% managed code in supporting local transactions. Local transactions are implemented within the .NET Framework and use the internal transaction manager of the underlying database.

The .NET Client supports savepoints through the Save() and Rollback(String) methods of the SequeLinkTransaction object.

If your application connects to only one database and you have no requirement to connect to another database, you should use local transactions. Local transactions are always faster than distributed transactions, which require additional logging and network I/O. In addition, local transactions provide the added benefit of increased security because the data provider does not need to call unmanaged code.

Using Distributed Transactions

The .NET Client supports distributed transactions through the Microsoft Distributed Transaction Coordinator (MS DTC). The MS DTC is provided through COM+ Services. These components are required to call some unmanaged code, which can affect the level of security available. If maintaining 100% managed code is important in your environment, do not use distributed transactions.

The .NET Client supports distributed transactions only when the following conditions are met:

- The Server version of SequeLink Client *for* .NET, including the optional MS DTC Support components, was installed.
 - DDTek.SequeLink.XA.dll must be in your Path.
 - DDTek.DTC.dll must be installed in the GAC.
- The application is written to Serviced Components. For a general discussion of using serviced components, refer to the Microsoft .NET Framework SDK documentation.

Applications that do not enlist in MS DTC coordinated transactions use only managed code, even if the MS DTC components are installed. Unmanaged code is called only if the application performs distributed transaction processing.

To use distributed transactions, you must include specific code in the namespace of your application, as shown in the following code fragment:

```
using System
using System.EnterpriseServices;
using DDTek.SequeLink
```

Distributed transactions are significantly slower than normal transactions due to the logging and network I/O necessary to communicate between all the components involved in the distributed transaction.

Because the .NET data provider is a managed data provider, you can still enjoy some of the benefits of the .NET Framework security when you use distributed transactions. The security afforded by the data provider, with the security defined in the underlying database, provides good protection when you use distributed transactions.

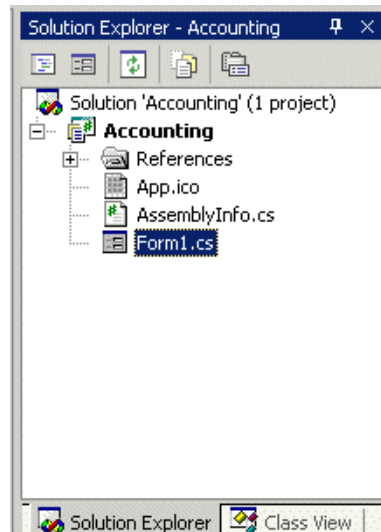
Connecting to a Database

Once the .NET Client is installed, you can connect from your application to your database with a connection string.

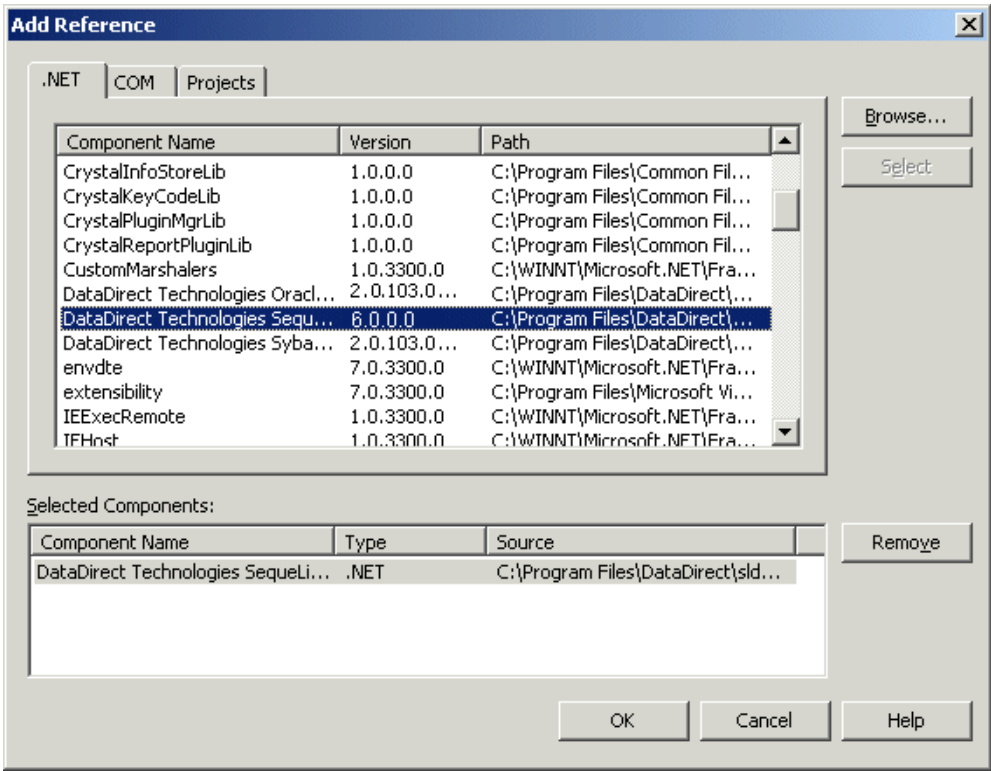
The following example illustrates connecting to the underlying Sybase database using the .NET Client from an application developed in Visual Studio .NET. If you are connecting using a different ADO.NET data provider or connecting from the command line, the specific details vary.

If you are using Visual Studio .NET:

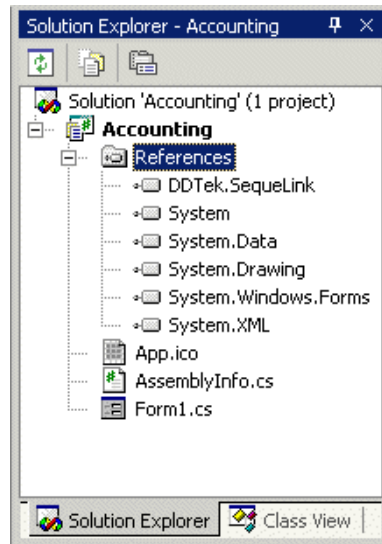
- 1 In the Solution Explorer, right-click **References**; then, click **Add Reference**.



- 2 Select the DataDirect Technologies Sequelink .NET Client in the component list.



- 3 Click **OK**. The Solution Explorer now includes a reference for the .NET Client.



- 4 Check the beginning of your application. If the data provider's namespace is not present, add it, as shown in the following code fragments:

C#

```
// Access SequeLink Server
using System.Data;
using DDTek.SequeLink;
```

Visual Basic

```
' Access SequeLink Server
Imports System.Data
Imports DDTek.SequeLink
```

- 5 Add the connection information for your server and exception handling code, as shown in the following code fragments.

C#

```
DBConn = new SequeLinkConnection("Host=sydney;
```

```

        Port=19998;User ID=test01;Password=test01;
        Database=test");

try
{
    DBConn.Open();
    Console.WriteLine ("Connection successful!");
}
// Display any exceptions
catch (SequeLinkException ex)
{
    // Connection failed
    Console.WriteLine (ex.Message);
    return;
}

```

Visual Basic

```

Dim Conn As New SequeLinkConnection("Host=bowhead;
Port=19996;User ID=test01;Password=test01;
Database=test")

Try
    'Open the connection
    Conn.Open()
    MessageBox.Show("Connection successful!")

Catch SLex As SequeLinkException
    'Connection failed
    MessageBox.Show(SLex.Message, "SequeLink Exception")

End Try

```

6 Close the connection.

C#

```

// Close the connection
DBConn.Close();

```

Visual Basic

```
'Close the connection  
Conn.Close()
```


10 Developing .NET Applications

This chapter provides information about developing .NET applications for the SequeLink environment including:

- ["Namespace" on page 389](#)
- ["Data Types" on page 390](#)
- ["Isolation Levels" on page 395](#)
- ["Threading" on page 395](#)
- ["Event Handling" on page 396](#)
- [".NET Public Objects/Interfaces Supported" on page 396](#)
- ["Setting .NET Security Permissions" on page 405](#)
- ["Error Handling" on page 406](#)
- ["Diagnostic Support" on page 408](#)
- ["Designing .NET Applications for Performance" on page 412](#)

Developers of data consumer applications must be familiar with the Microsoft .NET specification and object-oriented programming techniques.

Namespace

The namespace for the SequeLink .NET data provider is `DDTek.SequeLink`. When connecting to the SequeLink server, you use the `SequeLinkConnection` and `SequeLinkCommand` objects in the `DDTek.SequeLink` namespace.

The following C# code fragment shows how to include the SequeLink data provider's namespace in your applications:

```
// Access SequeLink Server
using System.Data;
using DDTek.SequeLink;
```

Data Types

The data types and isolation levels supported by the .NET data provider depend on the data store to which you are connecting.

Table 10-1 lists the data types supported by the .NET data provider and how they are mapped to the .NET Framework types. You can use the table to infer the data types that will be used when a DataSet is filled using a DataAdapter. This table also identifies the proper accessors for accessing the data when a DataReader object is used directly.

- The SequeLink data type column refers to the native type name.
- The SequeLinkDbType column refers to the SequeLink data provider's type enumeration. Generally, there is a one to one mapping between the native type and the SequeLinkDbType.
- The .NET Framework Type column refers to the base data types available in the framework.
- The .NET Framework Typed Accessor column refers to the method that must be used to access a column of this type when using a DataReader.

Table 10-1. Mapping of SequeLink Data Types

SequeLink Service Data Type	SequeLinkDbType	.NET Framework Type	.NET Framework Typed Accessor
BIGINT	Bigint	Int64	GetInt64()
BINARY	Binary	Byte[]	GetBytes()
BIT	Bit	Boolean	GetBoolean()
BLOB	Blob	Byte[]	GetBytes()
CHAR	Char	String	GetString() GetChars()

Table 10-1. Mapping of SequeLink Data Types (cont.)

SequeLink Service Data Type	SequeLinkDbType	.NET Framework Type	.NET Framework Typed Accessor
CLOB	Clob	String	GetString() GetChars()
DATE	Date	DateTime	GetDateTime()
DECIMAL	Decimal	Decimal	GetDecimal()
DOUBLE	Double	Double	GetDouble()
FLOAT	Double	Double	GetDouble()
GRAPHIC	Char	String	GetString() GetChars()
INTEGER	Int	Int32	GetInt32()
LONGVARBINARY	LongVarBinary	Byte[]	GetBytes()
LONGVARCHAR	LongVarChar	String	GetString() GetChars()
NUMBER	Decimal	Decimal	GetDecimal()
REAL	Single	Single	GetFloat()
SMALLINT	SmallInt	Int16	GetInt16()
TIME	Time	TimeSpan	GetValue()
TIMESTAMP	TimeStamp	DateTime	GetDateTime()
TINYINT	Byte	Byte[]	GetByte()
VARBINARY	VarBinary	Byte[]	GetBytes()
VARCHAR	VarChar	String	GetString() GetChars()
VARGRAPHIC	VarChar	String	GetString() GetChars()

Mapping Parameter Data Types

The type of the parameter is specific to each data provider. The .NET data provider must convert the parameter value to a native format before sending it to the server. The best way for an application to describe a parameter is to use the data provider-specific type enumeration.

In generic programming circumstances, the data provider-specific type may not be available. When no provider-specific DB type has been specified, the data type will be inferred from either the System.Data.DbType or from the .NET framework type of the parameter’s value.

The .NET data provider uses the following order when inferring the data type of a parameter:

- The data provider uses the provider-specific data type if it has been specified.
- The data provider infers the data type from the System.Data.DbType if it has been specified, but the provider-specific data type has not been specified.
- The data provider infers the data type from the .NET Framework type if neither the provider-specific data type nor the System.Data.DbType have been specified.

Table 10-2 shows how SequeLink infers its types if only the System.Data.DbType is specified.

Table 10-2. Mapping of the System.Data.DbTypes to SequeLinkDbTypes

System.Data.DbType	SequeLinkDbType
AnsiString	VarChar
AnsiStringFixedLength	Char
Binary	VarBinary

Table 10-2. Mapping of the System.Data.DbTypes to SequeLinkDbType *(cont.)*

System.Data.DbType	SequeLinkDbType
Boolean	Bit
Byte	Byte
Currency	Decimal
Date	Date
DateTime	TimeStamp
Decimal	Decimal
Double	Double
Guid	Not supported
Int16	SmallInt
Int32	Int
Int64	BigInt
Object	Not supported
Sbyte	Byte
Single	Single
String	VarChar
StringFixedLength	Char
Time	Time
UInt16	Int
UInt32	Decimal
UInt64	Decimal
VarNumeric	Not supported

Table 10-3 shows the mapping that the data provider uses to infer a data type if neither the provider-specific data type or the System.Data.DbType are provided.

<i>Table 10-3. Mapping .NET Framework Data Types to SequeLinkDbTypes</i>	
.NET Framework Type	SequeLinkDbType
Boolean	Bit
Byte	Byte
Byte[]	Varbinary
Char, Char[]	Not supported
DateTime	TimeStamp
Decimal	Decimal
Double	Double
Guid	Not supported
Int16	SmallInt
Int32	Int
Int64	BigInt
Object	Not supported
Single	Single
String	VarChar
Timespan	Time
UInt16	Int
UInt32	Decimal
UInt64	Decimal

Isolation Levels

The isolation levels supported by the .NET Client depend on the data store to which you are connecting. Isolation levels are set using the `BeginTransaction(IsolationLevel)` method of the `SequeLinkConnection` object.

See [Appendix B “Data Types and Isolation Levels”](#) on page 451 for database-specific information about data types and isolation levels.

Threading

The `SequeLinkConnection` object is thread-safe. Multiple `SequeLinkCommand` objects, each accessed on a separate thread, can simultaneously use a single connection.

Accessing other public and data provider-specific objects simultaneously on separate threads is not thread-safe.

Event Handling

The event handler receives an argument of type `SequeLinkInfoMessageEventArgs`, which contains data relevant to an event. See [“SequeLinkInfoMessageEventArgs Object” on page 402](#) for more information.

This event is defined as:

```
public event SequeLinkInfoMessageEventHandler InfoMessage;
```

Clients that want to process warnings and informational messages sent by the database server should create a `SequeLinkInfoMessageEventHandler` delegate to listen to this event.

The following code fragment defines a delegate that represents the method that handles the `InfoMessage` event of a `SequeLinkConnection` object:

```
[Serializable]  
public delegate void SequeLinkInfoMessageEventHandler(  
    object sender  
    SequeLinkInfoMessageEventArgs e  
);
```

where *sender* is the object that generated the event and *e* is a `SequeLinkInfoMessageEventArgs` object that describes the warning.

.NET Public Objects/Interfaces Supported

The .NET data provider supports the .NET public classes, interfaces, properties, and methods. The .NET data provider attaches the provider-specific prefix "SequeLink" to the names of the public objects, for example, `SequeLinkConnection` or

SequeLinkCommand. In addition, the .NET data provider supports provider-specific properties and methods.

The .NET data provider supports the public .NET objects, properties, and methods described in the following sections. See [Appendix G “.NET Code Examples” on page 577](#) for sample code that uses these objects.

SequeLinkCommand Object

In addition to the public properties of the Command object, the SequeLinkCommand object supports the properties described in [Table 10-4](#). The table includes the generic public properties of the Command object when provider-specific information supplements the standard descriptions.

For information about other properties and methods supported, refer to the data provider’s online help and the Microsoft .NET Framework SDK documentation.

Table 10-4. Properties of the SequeLinkCommand Object

Property	Description
ArrayBindCount	<p>Specifies the number of rows of parameters that will be used. The application must set this property before executing a command that uses parameter array binding. The count must equal the length of each of the arrays that is set for each parameter value.</p> <p>When set to 0 (the initial default), the application does not use parameter array binding.</p>

Table 10-4. Properties of the SequeLinkCommand Object (cont.)

Property	Description
ArrayBindStatus	<p>Enables the application to inspect the per row status after executing a command that uses parameter array binding. The property's type is an array of <code>OracleRowStatus</code>.</p> <p>Parameter array binding is performed as a single atomic operation. This means that if the operation succeeds, every entry will be set to OK; if the operation fails, none of the entries will be set to OK. The <code>OracleRowStatus</code> enumeration has the following possible values:</p> <ul style="list-style-type: none">■ OK. The operation succeeded. All entries are marked as OK.■ Failed. The operation failed. The data provider assigns this value to all status entries except for the row that caused the failure.■ SchemaViolation. When an operation fails, the data provider assigns this value to the row that caused the failure.
RowSetSize	<p>Limits the number of rows returned by any query executed on this Command object to the value specified at execute time. The data type for the Read-Write property is unsigned long.</p> <p>Valid values are 0 to 2147483647. When set to 0, the data provider does not limit the number of rows returned.</p> <p>The initial default is 0.</p>

SequeLinkCommandBuilder Object

The `SequeLinkCommandBuilder` object automatically generates single-table SQL commands that are used to reconcile changes made to a `DataSet` with the `SequeLink` server. A `SequeLinkCommandBuilder` object is always associated with a `SequeLinkDataAdapter` object.

Using a `CommandBuilder` object can have a negative effect on performance. Because of concurrency restrictions, the `CommandBuilder` does not generate efficient SQL statements. The end-user can often write more efficient update and delete

statements than those that the `CommandBuilder` generates. In addition, the `CommandBuilder` object registers itself as a listener for the `RowUpdating` and `RowUpdated` events of its `DataAdapter` object. This means that two events must be processed for every row that is updated.

In addition to the public properties of the `CommandBuilder` object, the `SequeLinkCommandBuilder` object supports the following property. For information about other properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

Table 10-5. Properties of the `SequeLinkCommandBuilder` Object

Property	Description
<code>AllowUpdateWithoutKey</code>	<p>Allows the use of the <code>SequeLinkCommandBuilder</code> on a table that has no primary key or unique index.</p> <p>When set to <code>True</code>, a <code>DataAdapter.Update</code> can succeed when a single SQL statement updates multiple rows in the underlying table. Normally, each <code>Update</code> or <code>Delete</code> statement executed on a <code>DataAdapter.Update</code> changes only one row.</p> <p>The initial default is <code>False</code>.</p>

SequeLinkConnection Object

In addition to the public properties of the `Connection` object, the `SequeLinkConnection` object supports the properties described in [Table 10-6](#). Some properties return the values specified for the corresponding connection string attribute (see ["Specifying Connection Options" on page 369](#) for information on the connection string attributes).

For information about other properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

Table 10-6. Properties of the SequeLinkConnection Object

Property	Description
ApplicationId	Specifies the application ID that identifies the client application to the SequeLink service. This attribute is only required when the SequeLink service you are connecting to has been configured to limit access to specific applications. For more information about using application IDs to limit access to the SequeLink services, refer to the <i>SequeLink Administrator's Guide</i> .
Host	Returns the value specified for the Host attribute in the connection string.
Port	Returns the value specified for the Port attribute in the connection string.
ServerDatasource	Returns the value specified for the ServerDatasource attribute in the connection string.
ServerVersion	Returns the name and version of the database that the object is connected to.

NOTE: To avoid dead connections being kept open on the database server, you must close or dispose every opened Connection object before it goes out of scope. Opened connections are not closed automatically when the memory is reclaimed by the garbage collector.

SequeLinkDataAdapter Object

The SequeLinkDataAdapter object uses SequeLinkCommand objects to execute SQL commands on the SequeLink Server database, to load the DataSet with data, and to reconcile the changed data in the DataSet to the database.

The SequeLinkDataAdapter object supports the public properties of the DataAdapter object and has no provider-specific properties. For information about the properties and methods

supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

SequeLinkDataReader Object

The SequeLinkDataReader object is a forward-only cursor that retrieves read-only records from a database. Performance is better than SequeLinkDataAdapter, but the result set cannot be modified.

For information about the properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

NOTE: To avoid statements or cursors being kept open on the database server, be sure to close or dispose each SequeLinkDataReader object as soon as you are finished reading with it.

SequeLinkError Object

The SequeLinkError object collects information relevant to errors and warnings generated by the SequeLink server.

For information about the properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

SequeLinkErrorCollection Object

The SequeLinkErrorCollection object is created by a SequeLinkException to contain all the errors generated by the SequeLink server.

The `SequeLinkErrorCollection` object supports the public properties of the `ErrorCollection` object, and has no provider-specific properties. For information about other properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

SequeLinkException Object

The `SequeLinkException` object is created and thrown when the `SequeLink` server returns an error. Exceptions generated by the data provider itself are returned as standard run time exceptions.

The `SequeLinkException` object supports the public properties of the `Exception` object, and has no provider-specific properties. The properties apply to the last error generated, if multiple errors exist. The application should check the `Count` property of the `SequeLinkErrorCollection` returned in the `Errors` property of this object to determine whether there are multiple errors.

For information about the properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

SequeLinkInfoMessageEventArgs Object

The `SequeLinkInfoMessageEventArgs` object is passed as an input to the `SequeLinkInfoMessageEventHandler` and contains information relevant to a warning generated by the `SequeLink` server. See [“Error Handling” on page 406](#) for an example of using `InfoMessage` delegates to retrieve warning information.

For information about `SequeLinkInfoMessageEventArgs`, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

SequeLinkParameter Object

The SequeLinkParameter object represents a parameter to a SequeLinkCommand. In addition to the public properties of the Parameter object, the SequeLinkParameter object supports the properties described in [Table 10-7](#).

For information about other properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

Table 10-7. Properties of the SequeLinkParameter Object

Property	Description
ArrayBindStatus	Determines whether any values in an array of SequeLinkParameterStatus entries should be bound as null. The SequeLinkParameterStatus enumeration contains the entry Null.
SequeLinkDbType	Specifies the data types of the SequeLink server. See "Data Types" on page 390 for more information about SequeLink data types.
Value	Gets or sets the value of the parameter. This property is specified as an array of values when array binding is enabled (see the ArrayBindCount property of the "SequeLinkCommand Object" on page 397). Each array's length must match the value of the ArrayBindCount property. When specifying the array's values for binary type columns, the data will actually be specified as byte[]. This is an array of arrays of bytes. The data provider anticipates a jagged array as such when using parameter array binding with parameters.

SequeLinkTrace Object

The SequeLinkTrace object is created by the application to debug problems during development. For your final application, be sure to remove references to the SequeLinkTrace object. Setting the properties in the SequeLinkTrace object overrides the

settings of the environment variables (see [“Specifying Connection Options”](#) on page 369).

See [“Using Environment Variables”](#) on page 409 for information about using environment variables.

The .NET data provider provides the option to create a Trace object to help application programmers debug problems during development. To maintain security, trace logs show passwords as five asterisks (*****).

The following code fragment creates a Trace object. All subsequent calls to the data provider will be traced to that file.

```
SequeLinkTrace MyTraceObject = new SequeLinkTrace();
MyTraceObject.TraceFile="C:\\MyTrace";
MyTraceObject.EnableTrace = 1;
```

[Table 10-8](#) provides the properties for the SequeLinkTrace object.

Table 10-8. Properties of the SequeLinkTrace Object

Property	Description
EnableTrace	When set to 1 or higher, enables tracing. When set to 0 (the initial default), tracing is disabled.
RecreateTrace	When set to 1, recreates the trace file each time the application restarts. When set to 0 (the initial default), the trace file is appended.
TraceFile	When DDTek_Enable_Trace is set to 1 or higher, this option specifies the path and name of the trace file. The initial default is \SequeLinkTrace.txt. If the file does not exist, the data provider creates it.

NOTE: Setting EnableTrace starts the tracing process. Therefore, you must define the property values for the trace file before setting EnableTrace. Once the trace processing starts, the values of TraceFile and RecreateTrace cannot be changed.

SequeLinkTransaction Object

The SequeLinkTransaction object implements the IDBTransaction interface as specified in the .NET Framework.

[Table 10-9](#) provides the methods used by the SequeLinkTransaction object to support savepoints. For information about other properties and methods supported, refer to the data provider's online help and the Microsoft .NET Framework SDK documentation.

Table 10-9. Methods of the SequeLinkTransaction Object

Method	Description
Save	Specifies the savepoint name, and creates a savepoint in the transaction that can be used to roll back a portion of the transaction.
Rollback(String)	Rolls back the specified transaction to a savepoint from a pending state.

Setting .NET Security Permissions

The .NET data provider implements security through the security permissions defined by the .NET Framework.

Code Access Permissions

The .NET data provider requires the FullTrust permission to be set in order to load and run. This requirement is due to underlying classes in System.Data that demand FullTrust for inheritance. All ADO.NET data providers require these classes to implement a DataAdapter.

Security Attributes

The .NET data provider is marked with the `AllowPartiallyTrustedCallers` attribute.

Error Handling

The following types of errors can occur when you are using the SequeLink Client *for* .NET:

- .NET errors
- SequeLink Client errors
- SequeLink Server errors
- Database errors

.NET Errors

The `SequeLinkError` object collects information relevant to errors and warnings generated by the database server. See [“SequeLinkError Object” on page 401](#) for more information.

See [“Retrieving Warning Information” on page 593](#) for a code example of how to return warning information.

The `SequeLinkException` object is created and thrown when the database server returns an error. Exceptions generated by the data provider are returned as standard run time exceptions. See [“SequeLinkException Object” on page 402](#) for more information.

ADO.NET Data Provider Errors

Errors generated by the .NET Client have different formats, depending on the cause and source of the problem. Formats include:

```
[SequeLink nnnn] Memory allocation error occurred.  
Invalid parameter type.
```

If a native error code is displayed, you can look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

SequeLink® Server Errors

An error generated by SequeLink Server has the following format:

```
DDTek.SequeLink.SequeLinkException:[SequeLink Server]  
message
```

For example:

```
DDTek.SequeLink.SequeLinkException:[SequeLink Server]  
Required user name is missing.
```

Use the native error code to look up details about the possible cause of the error. For a list of all error codes and messages, refer to the *SequeLink Troubleshooting Guide and Reference*.

Database Errors

An error generated by the database has the following format:

```
DDTek.SequeLink.SequeLinkException:[...] message
```

For example:

```
DDTek.SequeLink.SequeLinkException:[Oracle] ORA-00942:table  
or view does not exist.
```

Use the native error code to look up details about the possible cause of the error. For these details, refer to your database documentation.

Diagnostic Support

The data provider delivers advanced diagnostic capability:

- Ability to trace method calls
- PerfMon counters that let you tune connection information for your application

Tracing Method Calls

Tracing capability can be enabled either through environment variables or the provider-specific `SequeLinkTrace` class. The data provider traces the input arguments to all of its public method calls, as well as the outputs and returns from those methods (anything that a user could potentially call). Each call contains trace entries for entering and exiting the method.

During debugging, sensitive data can be read, even if it is stored as a private or internal variable and access is limited to the same

assembly. To maintain security, trace logs show passwords as five asterisks (*****).

Note, however, that if the Persist Security Info connection string option is set to true, the password will be displayed in clear text. See [“Using Connection Pooling” on page 364](#) for more information about connection strings.

Using Environment Variables

Using environment variables to enable tracing means that you do not have to modify your application. If you change the value of an environment variable, you must restart the application for the new value to take effect.

To enable and control tracing, set the following environment variables:

DDTek_Trace_File	Specifies the path and name of the trace file. The initial default is \SequeLinkTrace.txt.
DDTek_Recreate_Trace	When set to 1, re-creates the trace file each time the application restarts. When set to 0 (the initial default), the trace file is appended.
DDTek_Enable_Trace	When set to 1 or higher, enables tracing. When set to 0 (the initial default), tracing is disabled.

Using Static Methods

Some users may find that using static methods on the data provider's Trace class to be a more convenient way to enable tracing. The following C# code fragment uses static methods on the .NET Trace object to create a SequeLinkTrace class with a trace file named MyTrace.txt. The values set override the values

set in the environmental variables. All subsequent calls to the data provider will be traced to MyTrace.txt.

```
SequeLinkTrace.TraceFile="C:\\MyTrace.txt";
SequeLinkTrace.RecreateTrace = 1;
SequeLinkTrace.EnableTrace = 1;
```

The trace output has the following format:

```
<Correlation#> <Timestamp> <CurrentThreadName>
  <Object Address> <ObjectName.MethodName> ENTER (or EXIT)
    Argument #1 : <Argument#1 Value>
    Argument #2 : <Argument#2 Value>
    ...
  RETURN: <Method ReturnValue> // This line only exists for
EXIT
```

where:

Correlation# is a unique number that can be used to match up ENTER and EXIT entries for the same method call in an application.

Value is the hash code of an object appropriate to the individual function calls.

PerfMon Support

The Performance Monitor (PerfMon) utility in the Windows operating system allows you to record application parameters and review the results as a report or graph. You can also use Performance Monitor to identify the number and frequency of CLR exceptions in your applications.

The SequeLink *for* .NET data provider installs a set of PerfMon counters that let you tune and debug applications that use the data provider. The data provider's counters are located in the Performance Monitor under a category name, for example, SequeLink .NET Data Provider.

Table 10-10 describes the counters that you can use to tune connections for your application.

Table 10-10. PerfMon Counters

Counter	Description
Current # of Connection Pools	Returns the current number of pools associated with the process.
Current # of Pooled Connections	Returns the current number of connections in all pools associated with the process.
Current # of Pooled and Non-Pooled Connections	Returns the current number of pooled and non-pooled connections.
Peak # of Pooled Connections	Returns the highest number of connections in all connection pools since the process started.
Total # of Failed Connects	Returns the total number of attempts to open a connection that failed for any reason since the process started.
Total # of Failed Commands	Returns the total number of command executions that failed for any reason since the process started.

For information on using PerfMon and performance counters, refer to the Microsoft documentation library.

Designing .NET Applications for Performance

This section provides some tips for creating .NET applications that will perform efficiently.

Developing performance-oriented .NET applications is not easy. The .NET data providers do not throw exceptions to say that your code is running too slowly.

Selecting .NET Objects and Methods

The guidelines in this section will help you to optimize system performance when selecting and using .NET objects and methods.

Choosing Between a DataSet and a DataReader

A critical choice when designing your application is whether to use a DataSet or a DataReader. If you need to retrieve many records rapidly, use a DataReader. The DataReader object is fast, returning a fire hose of read-only data from the server, one record at a time. In addition, retrieving results with a DataReader requires significantly less memory than creating a DataSet. The DataReader does not allow random fetching, nor does it allow for updating the data. However, .NET data providers optimize their DataReaders for efficiently fetching large amounts of data.

In contrast, the DataSet object is a cache of disconnected data stored in memory on the client. In effect, it is a small database in itself. Because the DataSet contains all of the data that has been retrieved, you have more options in the way you can process the data. You can randomly choose records from within the DataSet and update, insert, and delete records at will. You can also manipulate relational data as XML. This flexibility provides some impressive functionality for any application, but comes with a

high cost in memory consumption. In addition to keeping the entire result set in memory, the DataSet maintains both the original and the changed data, which leads to even higher memory usage. Do not use DataSets with very large result sets because the scalability of the application will be drastically reduced.

Using Parameter Markers as Arguments to Stored Procedures

When calling stored procedures, always use parameter markers for the argument markers instead of using literal arguments.

.NET data providers can call stored procedures on the database server either by executing the procedure the same way as any other SQL query, or by optimizing the execution by invoking a Remote Procedure Call (RPC) directly into the database server. When you execute the stored procedure as a SQL query, the database server parses the statement, validates the argument types, and converts the arguments into the correct data types.

Remember that SQL is always sent to the database server as a character string, for example, "getCustName (12345)". In this case, even though the application programmer might assume that the only argument to getCustName is an integer, the argument is actually passed inside a character string to the server. The database server parses the SQL query, consults database metadata to determine the parameter contract of the procedure, isolates the single argument value 12345, then converts the string '12345' into an integer value before finally executing the procedure as a SQL language event.

Invoking an RPC inside the database server avoids the overhead of using a SQL character string. Instead, a .NET data provider constructs a network packet that contains the parameters in their native data type formats, and executes the procedure remotely.

To use stored procedures correctly, set the `CommandText` property of the `Command` object to the name of the stored procedure. Then, set the `CommandType` property of the command to `StoredProcedure`. Finally, pass the arguments to the stored procedure using parameter objects. Do not physically code the literal arguments into the `CommandText`.

Example 1

```
SequeLinkCommand DBCmd = new SequeLinkCommand("getCustName", Conn);  
SequeLinkDataReader = myDataReader;  
myDataReader = DBCmd.ExecuteReader();
```

In this example, the stored procedure cannot be optimized to use a server-side RPC. The database server must treat the SQL request as a normal language event which includes parsing the statement, validating the argument types, and converting the arguments into the correct data types before executing the procedure.

Example 2

```
SequeLinkCommand DBCmd = new SequeLinkCommand("getCustName", Conn);  
DBCmd.Parameters.Add("param1", SequeLinkDbType.Int, 10, "").Value =  
12345  
myDataReader.CommandType = CommandType.StoredProcedure;  
myDataReader = DBCmd.ExecuteReader();
```

In this example, the stored procedure can be optimized to use a server-side RPC. Because the application avoids literal arguments and calls the procedure by specifying all arguments as parameters, the .NET data provider can optimize the execution by invoking the stored procedure directly inside the database as an RPC. This example avoids SQL language processing on the database server and the execution time is greatly improved.

Avoiding the CommandBuilder Object

It is tempting to use a `CommandBuilder` object because it generates SQL statements and can save the developer time when coding a new application that uses `DataSets`. However, this shortcut can have a negative effect on performance. Because of concurrency restrictions, the `Command Builder` can generate highly inefficient SQL statements. For example, suppose you have a table called `EMP`, an 8-column table with simple employee records. A `CommandBuilder` would generate the following update statement:

```
CommandText: "UPDATE EMP SET EMPNO = ?, ENAME = ?, JOB = ?, MGR = ?, HIREDATE = ?, SAL = ?, COMM = ?, DEPT = ? WHERE ( (EMPNO = ?) AND (ENAME = ?) AND (JOB = ?) AND ((MGR IS NULL AND ? IS NULL) OR (MGR = ?)) AND (HIREDATE = ?) AND (SAL = ?) AND ((COMM IS NULL AND ? IS NULL) OR (COMM = ?)) AND (DEPT = ?) )"
```

The end-user can often write much more efficient `UPDATE` and `DELETE` statements than those that the `CommandBuilder` generates. For example, a programmer who knows the underlying database schema and that the `EMPNO` column of the `EMP` table is the primary key for the table, can code the same `UPDATE` statement as follows:

```
UPDATE EMP SET EMPNO = ?, ENAME = ?, JOB = ?, MGR = ?,  
HIREDATE = ?, SAL = ?, COMM = ?, DEPT = ? WHERE EMPNO = ?
```

This statement will run much more efficiently on the database server than the statement generated by the `CommandBuilder`.

Another drawback is also implicit in the design of the `CommandBuilder` object. The `CommandBuilder` must generate statements at runtime. Each time a `DataAdapter.Update` method is called, the `CommandBuilder` must analyze the contents of the result set and generate `UPDATE`, `INSERT`, and `DELETE` statements for the `DataAdapter`. When the programmer explicitly specifies the `UPDATE`, `INSERT`, and `DELETE` statements for the `DataAdapter`, this extra processing time is avoided.

Designing .NET Applications

The guidelines in this section will help you to optimize system performance when designing .NET applications.

Using Connection Pooling

Connecting to a database is the single slowest operation inside a data-centric application. That's why connection management is important to application performance. Optimize your application by connecting once and using multiple statement objects, instead of performing multiple connections. Avoid connecting to a data source after establishing an initial connection.

Connection pooling is part of the .NET data provider. Connection pooling lets you *reuse* connections. Closing connections does not close the physical connection to the database. When an application requests a connection, an active connection is reused, thus avoiding the network I/O needed to create a new connection.

Pre-allocate connections. Decide which connection strings you will need to meet your needs. Remember that each unique connection string creates a new connection pool.

```
host=norman;Port=19996;  
User ID=test01;Password=test01;"host=norman;Port=19996;  
host=norman;Port=19996;  
User ID=test01;Password=test01;
```

Once created, connection pools are not destroyed until the active process ends or the connection lifetime is exceeded. Maintenance of inactive or empty pools involves minimal system overhead.

Connection and statement handling should be addressed before implementation. Spending time and thoughtfully handling connection management improves application performance and maintainability.

Opening and Closing Connections

Open connections just before they are needed. Opening them earlier than necessary decreases the number of connections available to other users and can increase the demand for resources.

To keep resources available, explicitly Close the connection as soon as it is no longer needed. If you wait for the garbage collector to implicitly clean up connections that go out of scope, the connections will not be returned to the connection pool immediately, tying up resources that are not actually being used.

Close connections inside a finally block. Code in the finally block always runs, even if an exception occurs. This guarantees explicit closing of connections. For example:

```
try
{
    DBConn.Open();
    ... // Do some other interesting work
}
catch (Exception ex)
{
    // Handle exceptions
}
finally
{
    // Close the connection
    if (DBConn != null)
        DBConn.Close();
}
```

If you are using connection pooling, opening and closing connections is not an expensive operation. Using the Close() method of the data provider's Connection object adds or returns the connection to the connection pool. Remember, however, that closing a connection automatically closes all DataReader objects associated with the connection.

Managing Commits in Transactions

Committing transactions is slow due to the result of disk input/output and, potentially, network input/output. Always start a transaction after connecting; otherwise, you are in autocommit mode.

What does a commit actually involve? The database server must flush back to disk every data page that contains updated or new data. This is usually a sequential write to a journal file, but nonetheless, is a disk input/output. By default, Autocommit is on when connecting to a data source. Autocommit mode usually impairs performance because of the significant amount of disk input/output needed to commit every operation.

Furthermore, some database servers do not provide an autocommit mode natively. For this type of server, the .NET data provider must explicitly issue a COMMIT statement and a BEGIN TRANSACTION for every operation sent to the server. In addition to the large amount of disk input/output required to support autocommit mode, a performance penalty is paid for up to three network requests for every statement issued by an application.

The following code fragment starts a transaction for SequeLink:

```
SequeLinkConnection MyConn = new SequeLinkConnection
                                ("Connection String info");
MyConn.Open()

// Start a transaction
SequeLinkTransaction TransId = MyConn.BeginTransaction();

// Enlist a command in the current transaction
SequeLinkCommand SequeLinkToDS = new SequeLinkCommand();
SequeLinkToDS.Transaction = TransId;
...
// Continue on and do more useful work in the
// transaction
```

Although using transactions can help application performance, do not take this tip too far. Leaving transactions active can reduce throughput by holding locks on rows for long times, preventing other users from accessing the rows. Commit transactions in intervals that allow maximum concurrency.

Choosing the Right Transaction Model

Many systems support distributed transactions; that is, transactions that span multiple connections. Distributed transactions are at least four times slower than normal transactions due to the logging and network input/out needed to communicate between all the components involved in the distributed transaction (the .NET data provider, the transaction monitor, and the database system).

Distributed transactions should be used only when transactions must span multiple DBMSs or multiple servers. Unless they are required, avoid using distributed transactions. Instead, use local transactions whenever possible.

Using Commands that Retrieve Little or No Data

Commands such as INSERT, UPDATE and DELETE do not return data. Use these commands with ExecuteNonQuery method of the Command object. Although you can successfully execute these commands using the ExecuteReader method, the .NET Data Provider will properly optimize the database access for INSERT, UPDATE, and DELETE statements only through the ExecuteNonQuery method.

The following example shows how to insert a row into the EMPLOYEE table using ExecuteNonQuery:

```
DBConn.Open();  
DBTxn = DBConn.BeginTransaction();
```

```
// Set the Connection property of the Command object
DBCmd.Connection = DBConn;
// Set the text of the Command to the INSERT statement
DBCmd.CommandText = "INSERT into EMPLOYEE VALUES
(15, 'HAYES', 'ADMIN', 6, " +
    "'17-APR-2002', 18000, NULL, 4)";
// Set the transaction property of the Command object
DBCmd.Transaction = DBTxn;
// Execute the statement with ExecuteNonQuery, because we are not
// returning results
DBCmd.ExecuteNonQuery();
// Now commit the transaction
DBTxn.Commit();

// Close the connection
DBConn.Close();
```

Use the `ExecuteScalar` method of the Command object to return a single value, such as a sum or a count, from the database. The `ExecuteScalar` method returns only the value of the first column of the first row of the result set. Once again, you could use the `ExecuteReader` method to successfully execute such queries, but by using the `ExecuteScalar` method, you tell the .NET data provider to optimize for a result set that consists of a single row and a single column. By doing so, the data provider can avoid a lot of overhead and improve performance. The following example shows how to retrieve the count of a group:

```
// Retrieve the number of employees who make more than $50000
// from the EMPLOYEE table

// Open connection to Sybase database
SequeLinkConnection Conn;
Conn = new SequeLinkConnection("host=norman;Port=19996;
User ID=test01;Password=test01;");
Conn.Open();

// Make a command object
SequeLinkCommand salCmd = new SequeLinkCommand("SELECT count(sal) FROM" +
    "EMPLOYEES WHERE sal>'50000'", Conn);
```

```

try
{
    int count = (int)salCmd.ExecuteScalar();
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
Conn.Close();

```

Using Commands Multiple Times

Choosing whether to use the `Command.Prepare` method can have a significant positive (or negative) effect on query execution performance. The `Command.Prepare` method tells the underlying data provider to optimize for multiple executions of statements that use parameter markers. Note that it is possible to `Prepare` any command regardless of which execution method is used (`ExecuteReader`, `ExecuteNonQuery`, or `ExecuteScalar`).

Consider the case where a .NET data provider implements `Command.Prepare` by creating a stored procedure on the server that contains the prepared statement. Creating stored procedures involves substantial overhead, but the statement can be executed multiple times. Although creating stored procedures is performance-expensive, execution of that statement is minimized because the query is parsed and optimization paths are stored at create procedure time. Applications that execute the same statement multiples times can benefit greatly from calling `Command.Prepare` and then executing that command multiple times.

However, using `Command.Prepare` for a statement that is executed only once results in unnecessary overhead. Furthermore, applications that use `Command.Prepare` for large single execution query batches exhibit poor performance.

Similarly, applications that either always use `Command.Prepare` or never use `Command.Prepare` do not perform as well as those that use a logical combination of prepared and unprepared statements.

Using Native Managed Providers

Bridges into unmanaged code, that is, code outside the .NET environment, adversely affect performance. Calling unmanaged code from managed code causes the CLR (Common Language Runtime) to make additional checks on calls to the unmanaged code, which impacts performance.

The .NET CLR is a very efficient and highly tuned environment. By using 100% managed code so that your .NET assemblies run inside the CLR, you can take advantage of the numerous built-in services to enhance the performance of your managed application and your staff. The CLR provides automatic memory management, so developers don't have to spend time debugging memory leaks. Automatic lifetime control of objects includes garbage collection, scalability features, and support for side-by-side versions. In addition, the .NET Framework security enforces security restrictions on managed code that protects the code and data from being misused or damaged by other code. An administrator can define a security policy to grant or revoke permissions on an enterprise, a machine, an assembly, or a user level.

However, many .NET data provider architectures must bridge outside the CLR into native code to establish network communication with the database server. The overhead and processing required to enter this bridge is slow in the current version of the CLR.

Depending on your architecture, you may not realize that the underlying .NET data provider is incurring this security risk and performance penalty. Be careful when choosing a .NET data provider that advertises itself as a 100% or pure managed code

data provider. If the "Managed Data Provider" requires unmanaged database clients or other unmanaged pieces, then it is not a 100% managed data access solution. Only a very few vendors produce true managed code providers that implement their entire stack as a managed component.

Retrieving Data

To retrieve data efficiently, return only the data that you need, and choose the most efficient method of doing so. The guidelines in this section will help you to optimize system performance when retrieving data with .NET applications.

Retrieving Long Data

Unless it is necessary, applications should not request long data because retrieving long data across a network is slow and resource-intensive. Remember that when you use a DataSet, all data is retrieved from the data source, even if you never use it.

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 .NET data provider (that is, some applications use syntax such as `SELECT * FROM <table name> . . .`). If the select list contains long data, then some data providers must retrieve that data at fetch time even if the application does not bind the long data in the result set. When possible, the designer should attempt to implement a method that does not retrieve all columns of the table.

Most users don't want to see long data. If the user does want to see these result items, then the application can query the database again, 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.

Consider a query such as `SELECT * FROM Employees WHERE SSID = '999-99-2222'`. An application might only want to retrieve this employee's name and address. But, remember that a .NET data provider cannot tell which result columns an application might be trying to retrieve when the query is executed. A data provider only knows that an application can request any of the result columns. When the .NET data provider processes the fetch request, it will most likely return at least one, if not more, result rows across the network from the database server. In this case, a result row will contain all the column values for each row — including an employee picture if the Employees table happens to contain such a column. Limiting the select list to contain only the name and address columns results in decreased network traffic and a faster performing query at runtime.

Reducing the Size of Data Retrieved

To reduce network traffic and improve performance, you can reduce the size of any data being retrieved to some manageable limit by using a database-specific command. For example, an Oracle data provider might let you limit the number of bytes of data the connection uses to fetch multiple rows. A Sybase data provider might let you limit the number of bytes of data that can be returned from a single IMAGE column in a result set. For example, with Microsoft SQL Server and Sybase ASE, you can issue `Set TEXTSIZE n` on any connection, where *n* sets the maximum number of bytes that will ever be returned to you from any TEXT or IMAGE column.

In addition, be careful to return only the rows you need. If you return five columns when you only need two columns, performance is decreased, especially if the unnecessary rows include long data.

Especially when using a DataSet, be sure to use a WHERE clause with every SELECT statement to limit the amount of data that will be retrieved. Even when you use a WHERE clause, a SELECT statement that does not adequately restrict the request could

return hundreds of rows of data. For example, if you want the complete row of data from the EMPLOYEE table for each manger hired in recent years, you might be tempted to issue the following statement and then, in your application code, filter out the rows who are not managers:

```
SELECT * FROM EMPLOYEE WHERE hiredate > 2000
```

However, suppose the EMPLOYEE table contains a PHOTOGRAPH column. Retrieving all the extra rows could be extremely expensive. Let the database filter them for you and avoid having all the extra data that you don't need sent across the network. A better request further limits the data returned and improves performance:

```
SELECT * FROM EMPLOYEE WHERE hiredate > 2003 AND job_title=
'Manager'
```

Choosing the Right Data Type

Advances in processor technology brought significant improvements to the way that operations such as floating-point math are handled; however, retrieving and sending certain data types are still expensive when the active portion of your application will not fit into on-chip cache. When you are working with data on a large scale, it is still important to select the data type that can be processed most efficiently.

For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

Processing time is shortest for character strings, followed by integers, which usually require some conversion or byte ordering. Processing floating-point data and timestamps is at least twice as slow as integers.

Updating Data

This section provides general guidelines to help you to optimize system performance when updating data in databases.

Synchronizing Changes Back to the Data Source

The following example shows the application flow for updating a `DataSet` using Oracle's `Rowid` as the update mechanism:

```
// Create the DataAdapter and DataSets
SequeLinkCommand DbCmd = new SequeLinkCommand
("SELECT rowid, deptid, deptname FROM department", DBConn);

myDataAdapter = new SequeLinkDataAdapter();
myDataAdapter.SelectCommand = DbCmd;
myDataAdapter.Fill(myDataSet, "Departments");

// Build the Update rules
// Specify how to update data in the data set
myDataAdapter.UpdateCommand = new
SequeLinkCommand("Update department set deptname = ? ", deptid = ?
" +
    "WHERE rowid =?", DBConn);

// Bind parameters
myDataAdapter.UpdateCommand.Parameters.Add
    ("param1", SequeLinkDbType.VarChar, 100, "deptname");
myDataAdapter.UpdateCommand.Parameters.Add("param2",
    SequeLinkDbType.Number, 4, "deptid");
myDataAdapter.UpdateCommand.Parameters.Add("param3",
    SequeLinkDbType.Number, 4, "rowid");
```

In this example, performance of the queries on the Oracle server improves because the `WHERE` clause includes only the `rowid` as a search condition.

For More Information

Microsoft provides extensive documentation about ADO.NET on its World Wide Web site, including the following information:

- *Using .NET Providers to Access Data*
[http://msdn2.microsoft.com/en-us/library/s7ee2dwt\(vs.71\).aspx](http://msdn2.microsoft.com/en-us/library/s7ee2dwt(vs.71).aspx)
- *Transaction Control: Building Distributed Transactions with .NET*
<http://msdn2.microsoft.com/en-us/library/ms978457.aspx>
- *Writing Serviced Components*
[http://msdn2.microsoft.com/en-us/library/3x7357ez\(vs.71\).aspx](http://msdn2.microsoft.com/en-us/library/3x7357ez(vs.71).aspx)
- *Creating and Using DataSets*
[http://msdn2.microsoft.com/en-us/library/ss7fbaez\(vs.71\).aspx](http://msdn2.microsoft.com/en-us/library/ss7fbaez(vs.71).aspx)
- *XML and the DataSet*
[http://msdn2.microsoft.com/en-us/library/84sxtbxh\(vs.71\).aspx](http://msdn2.microsoft.com/en-us/library/84sxtbxh(vs.71).aspx)

Part 5: Reference

This part contains the following appendixes:

- [Appendix A “SQL Escape Sequences” on page 431](#) describes the scalar functions supported for SequeLink. Your data store may not support all these functions.
- [Appendix B “Data Types and Isolation Levels” on page 451](#) lists the data types and isolation levels supported for each data store supported by SequeLink.
- [Appendix C “JDBC Support” on page 489](#) provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- [Appendix D “JDBC Connection Pool Manager” on page 541](#) describes how your application can use connection pooling through the DataDirect Connection Pool Manager.
- [Appendix E “Troubleshooting Using DataDirect Spy™” on page 551](#) provides information to help you troubleshoot ODBC applications.
- [Appendix F “Developing ODBC Applications for Internationalization” on page 557](#) provides an overview of how to design your applications for internationalization, and provides the valid values for the IANAAppCodePage connection string attribute.
- [Appendix G “.NET Code Examples” on page 577](#) includes code examples of typical database access tasks in ADO.NET.

A SQL Escape Sequences

Language features, such as outer joins and scalar function calls, are commonly implemented by database systems. The syntax for these features is often database-specific, even when a standard syntax has been defined. ODBC and JDBC define escape sequences that contain standard syntaxes for the language features in [Table A-1](#).

NOTE: The .NET data provider supports ODBC/JDBC SQL escape sequences, except where otherwise noted.

Table A-1. Language Features

Language Feature	SequeLink Client <i>for</i> ODBC	SequeLink Client <i>for</i> JDBC	SequeLink Client <i>for</i> .NET
Date, time, and timestamp literals	X	X	X
Scalar functions such as numeric, string, and date type conversion functions	X	X	X
LIKE predicate escape characters	X	X	
Outer joins	X	X	X
Procedure call escape sequences	X	X	

The escape sequence used by JDBC is:

```
{extension}
```

The escape sequence is recognized and parsed by the driver or data provider, which replaces the escape sequences with data store-specific grammar.

The escape sequence is recognized and parsed by the driver, which replaces the escape sequences with data store-specific grammar.

The escape sequence is recognized and parsed by the data provider, which replaces the escape sequences with data store-specific grammar.

Date, Time, and Timestamp Escape Sequences

The escape sequence for date, time, and timestamp literals is:

```
{literal-type 'value'}
```

where *literal-type* is one of the following:

literal-type	Description	Value Format
d	Date	yyy-mm-dd
t	Time	hh:mm:ss
ts	Timestamp	yyyy-mm-dd hh:mm:ss[.f...]

Example:

```
UPDATE Orders SET OpenDate={d '1995-01-15'}  
WHERE OrderID=1023
```

Scalar Functions

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

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


where *scalar-function* is a scalar function supported by the ODBC driver, JDBC driver, and .NET data provider, as shown in the following tables.

scalar-function

Example:

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

Table A-2. Scalar Functions Supported on DB2

Data Store	String Functions	Numeric Functions	Time/date Functions	System Functions
DB2 UDB on z/OS	CHAR CONCAT DIFFERENCE INSERT LCASE LEFT LENGTH LOCATE LOCATE_2 LTRIM REPEAT REPLACE RIGHT RTRIM SOUNDEX SPACE SUBSTRING UCASE	ABS ACOS ASIN ATAN ATAN2 CEILING COS DEGREES EXP FLOOR LOG LOG10 MOD PI POWER RADIANS RAND ROUND SIGN SIN SQRT TAN TRUNCATE	CURDATE CURRENT_DATE CURTIME DAYOFMONTH DAYOFWEEK DAYOFYEAR HOUR MINUTE MONTH NOW SECOND WEEK YEAR	DBNAME IFNULL USERNAME

Table A-2. Scalar Functions Supported on DB2 *(cont.)*

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
DB2 UDB on Linux/UNIX/ Windows	ASCII	ABS	CURDATE	DATABASE
	CHAR	ACOS	CURRENT_DATE	IFNULL
	CONCAT	ASIN	CURTIME	USERNAME
	DIFFERENCE	ATAN	DAYNAME	
	INSERT	ATAN2	DAYOFMONTH	
	LCASE	CEILING	DAYOFWEEK	
	LEFT	COS	DAYOFYEAR	
	LENGTH	COT	HOUR	
	LOCATE	DEGREES	MINUTE	
	LOCATE_2	EXP	MONTH	
	LTRIM	FLOOR	MONTHNAME	
	REPEAT	LOG	NOW	
	REPLACE	LOG10	QUARTER	
	RIGHT	MOD	SECOND	
	RTRIM	PI	TIMESTAMPADD	
	SOUNDEX	POWER	TIMESTAMPDIFF	
	SPACE	RADIANS	WEEK	
	SUBSTRING	RAND	YEAR	
	UCASE	ROUND		
		SIGN		
		SIN		
		SQRT		
		TAN		
		TRUNCATE		

Table A-3. Scalar Functions Supported on Informix

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Informix	BIT_LENGTH CHAR_LENGTH CONCAT LENGTH LTRIM RTRIM STR_LENGTH	ABS ACOS ASIN ATAN ATAN2 COS COT EXP FLOOR LOG LOG10 MOD POWER ROUND SQRT TAN TRUNCATE	CURDATE CURRENT DATE CURTIME DAYOFMONTH DAYOFWEEK MONTH NOW QUARTER YEAR	DBNAME USERNAME

Table A-4. Scalar Functions Supported on Microsoft SQL Server

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Microsoft SQL Server	ASCII	ABS	CURDATE	DBNAME
	BITLENGTH	ACOS	CURRENT_DATE	IFNULL
	CHAR	ASIN	CURRENT_TIME	USERNAME
	CONCAT	ATAN	CURRENT_	
	DIFFERENCE	ATAN2	TIMESTAMP	
	INSERT	CEILING	CURTIME	
	LCASE	COS	DAYOFMONTH	
	LEFT	COT	DAYOFWEEK	
	LENGTH	DEGREES	DAYOFYEAR	
	LOCATE	EXP	DAYNAME	
	LOCATE2	FLOOR	EXTRACT	
	LTRIM	LOG	HOUR	
	OCTET_LENGTH	LOG10	MINUTE	
	REPEAT	MOD	MONTH	
	REPLACE	PI	MONTHNAME	
	RIGHT	POWER	NOW	
	RTRIM	RADIANS	QUARTER	
	SOUNDEX	RAND	SECOND	
	SPACE	ROUND	TIMESTAMPADD	
	SUBSTRING	SIGN	TIMESTAMPDIFF	
	UCASE	SIN	WEEK	
		SQRT	YEAR	
		TAN		
		TRUNCATE		

Table A-5. Scalar Functions Supported on Oracle

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Oracle	ASCII BIT_LENGTH CHAR CONCAT INSERT LCASE LEFT LENGTH LOCATE LOCATE2 LTRIM OCTET_LENGTH REPEAT REPLACE RIGHT RTRIM SOUNDEX SPACE SUBSTRING UCASE	ABS ACOS ASIN ATAN ATAN2 CEILING COS COT EXP FLOOR LOG LOG10 MOD PI POWER ROUND SIGN SIN SQRT TAN TRUNCATE	CURDATE CURRENT_DATE CURRENT_TIME CURRENT TIMESTAMP DAYOFMONTH DAYOFWEEK DAYOFYEAR DAYNAME HOUR MINUTE MONTH MONTHNAME NOW QUARTER SECOND TIMESTAMP_ADD TIMESTAMP_DIFF WEEK YEAR	IFNULL USER USERNAME

Table A-6. Scalar Functions Supported on Sybase

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Sybase	ASCII	ABS	CURDATE	DATABASE
	CHAR	ACOS	CURRENT_DATE	DBNAME
	CONCAT	ASIN	DAYOFMONTH	IFNULL
	DIFFERENCE	ATAN	DAYOFWEEK	USER
	INSERT	ATAN2	DAYOFYEAR	USERNAME
	LCASE	CEILING	DAYNAME	
	LEFT	COS	HOUR	
	LENGTH	COT	MINUTE	
	LOCATE	EXP	MONTH	
	LTRIM	FLOOR	MONTHNAME	
	REPEAT	LOG	NOW	
	RIGHT	LOG10	QUARTER	
	RTRIM	MOD	SECOND	
	SOUNDEX	NUM_DEGREES	TIMESTAMPADD	
	SPACE	NUM_RADIANS	TIMESTAMPDIFF	
	SUBSTRING	PI	WEEK	
	UCASE	POWER	YEAR	
		RADIANS		
		RAND		
		ROUND		
		SIGN		
		SIN		
		SQRT		
		TAN		
		TRUNCATE		

String Functions

[Table A-7](#) lists string functions. The following arguments can be used with these functions:

- *string_exp* can be a column name, a string literal, or the result of another scalar function, where the underlying data type is SQL_CHAR or SQL_WCHAR, SQL_VARCHAR or SQL_WVARCHAR, or SQL_LONGVARCHAR or SQL_WLONGVARCHAR.
- *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 the character 1. Character string literals must be enclosed by single quotation marks.

Table A-7. Scalar String Functions

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

Table A-7. Scalar String Functions (cont.)

Function	Returns
CHARACTER_LENGTH(<i>string_exp</i>)	The length, in characters, of the string expression, when the string expression is a character data type; otherwise, the length, in bytes, of the string expression (the lowest integer that is 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 indicating the difference between the values returned by the SOUNDEX function for <i>string_exp1</i> and <i>string_exp2</i> .
INSERT(<i>string_exp1</i> , <i>start</i> , <i>length</i> , <i>string_exp2</i>)	A string where <i>length</i> characters have been deleted from <i>string_exp1</i> beginning at <i>start</i> and where <i>string_exp2</i> has been inserted into <i>string_exp</i> beginning at <i>start</i> .
LCASE(<i>string_exp</i>)	Uppercase characters in <i>string_exp</i> converted to lowercase.
LEFT(<i>string_exp</i> , <i>count</i>)	The <i>count</i> of characters of <i>string_exp</i> .
LENGTH(<i>string_exp</i>)	The number of characters in <i>string_exp</i> , excluding trailing blanks and the string termination character.
LOCATE(<i>string_exp1</i> , <i>string_exp2</i> [, <i>start</i>])	The starting position of the first occurrence of <i>string_exp1</i> in <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 <i>start</i> . The first character position in <i>string_exp2</i> is indicated by 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>)	The length, in bytes, of the string expression. The result is the lowest integer that is not less than the number of bits divided by 8.

Table A-7. Scalar String Functions (cont.)

Function	Returns
POSITION(<i>character_exp</i> IN <i>character_exp</i>)	The position of the first character expression in the second character expression. The result is a 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 A-8 lists numeric functions. The following arguments can be used with numeric functions:

- *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 A-8. 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 A-8. 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 A-9 lists date and time functions. The following arguments can be used with the date and time functions:

- *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 A-9. Scalar Time and Date Functions

Function	Returns
CURDATE()	Current date as a date value.
CURRENT_DATE()	Current date.
CURRENT_TIME[(<i>time-precision</i>)]	Current local time. The <i>time-precision</i> argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[(<i>timestamp-precision</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.
CURTIME()	Current local time as a time value.
DAYNAME(<i>date_exp</i>)	Character string containing a data-source-specific name of the day for the day portion of <i>date_exp</i> .

Table A-9. Scalar Time and Date Functions (cont.)

Function	Returns
DAYOFMONTH(<i>date_exp</i>)	Day of the month in <i>date_exp</i> as an integer value (1–31).
DAYOFWEEK(<i>date_exp</i>)	Day of the week in <i>date_exp</i> as an integer value (1–7).
DAYOFYEAR(<i>date_exp</i>)	Day of the year in <i>date_exp</i> as an integer value (1–366).
EXTRACT(<i>extract-field</i> FROM <i>extract-source</i>)	<p><i>Extract-field</i> portion of the <i>extract-source</i>. The <i>extract-source</i> argument is a datetime or interval expression. The <i>extract-field</i> argument can be one of the following keywords:</p> <p>YEAR MONTH DAY HOUR MINUTE SECOND</p> <p>The precision scale of the returned value is 0 unless SECOND is specified, in which case, the scale is not less than the fractional seconds precision of the <i>extract-source</i> field.</p>
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>time_exp</i> as an integer value (0–59).

Table A-9. Scalar Time and Date Functions *(cont.)*

Function	Returns
<code>TIMESTAMPADD(interval, integer_exp, time_exp)</code>	<p>Timestamp calculated by adding <i>integer_exp</i> intervals of type <i>interval</i> to <i>time_exp</i>. <i>interval</i> can be one of the following values:</p> <p>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</p> <p>Fractional seconds are expressed in billionths of a second.</p>
<code>TIMESTAMPDIFF(interval, time_exp1, time_exp2)</code>	<p>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 value as <i>TIMESTAMPADD</i>. Fractional seconds are expressed in billionths of a second.</p>
<code>WEEK(date_exp)</code>	<p>Week of the year in <i>date_exp</i> as an integer value (1–53).</p>
<code>YEAR(date_exp)</code>	<p>Year in <i>date_exp</i>. The range is data-source dependent.</p>

System Functions

Table A-10 lists system functions.

Table A-10. Scalar System Functions

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

Like Predicate Escape Characters

In a LIKE predicate, the percent sign (%) matches zero or more of any character and the underscore (_) matches any one character. To match an actual percent sign or underscore in a LIKE predicate, an escape character must precede the % or _. The escape sequence that defines the LIKE predicate escape character is:

```
{escape 'escape-character'}
```

where *escape-character* is any character supported by the data source.

Example:

```
SELECT Name FROM Customers
WHERE Name LIKE '\%AAA%' {escape '\'}
```

Returns all the customers for which the name starts with "%AAA".

Outer Join Escape Sequences

ODBC and JDBC support the SQL92 left, right, and full outer join syntax. The escape sequence for outer joins is:

```
JDBC ODBC {oj outer-join}
```

where *outer-join* is:

```
table-reference {LEFT | RIGHT | FULL} OUTER JOIN  
{table-reference | outer-join} ON search-condition
```

where *table-reference* is a table name, and *search-condition* is the join condition you want to use for the tables.

Example:

```
SELECT Customers.CustID, Customers.Name, Orders.OrderID,  
Orders.Status  
FROM {oj Customers LEFT OUTER JOIN  
Orders ON Customers.CustID=Orders.CustID}  
WHERE Orders.Status='OPEN'
```

[Table A-11](#) lists the outer join escape sequences supported by SequeLink for each data store.

Table A-11. Outer Join Escape Sequences Supported

Data Store	Outer Join Escape Sequences
DB2 on z/OS	Left outer joins Right outer joins Full outer joins Nested outer joins Inner outer joins
DB2 on Windows and UNIX	Left outer joins Right outer joins Full outer joins Nested outer joins Unordered outer joins Inner outer joins

Table A-11. Outer Join Escape Sequences Supported (cont.)

Data Store	Outer Join Escape Sequences
Informix	Left outer joins Unordered outer joins
Microsoft SQL Server	Left outer joins Right outer joins Full outer joins Nested outer joins

Procedure Call Escape Sequences

A procedure is an executable object stored in the data store. Generally, it is one or more SQL statements that have been precompiled.

Procedure Call Escape Sequences

A procedure is an executable object stored in the data store. Generally, it is one or more SQL statements that have been precompiled.

The ODBC and JDBC drivers use the following escape sequence for calling a procedure:

```
{[?]=call procedure-name([parameter][, [parameter]]...)}
```

where:

procedure-name is the name of a stored procedure. You can call stored procedures with or without the schema name qualification.

parameter is a stored procedure parameter.

The ODBC JDBC driver uses the following escape sequence for calling a procedure:

```
{[?]=call procedure-name[(parameter [, parameter]...)]}
```

where:

procedure-name is the name of a stored procedure. You can call stored procedures with or without the schema name qualification.

parameter is a stored procedure parameter.

B Data Types and Isolation Levels

This appendix lists the data types and isolation levels supported for each data store supported by SequeLink.

Supported Data Types

Retrieving and sending certain data types can be expensive. Advances in processor technology brought significant improvements to the way that operations such as floating-point math are handled; however, retrieving and sending certain data types are still expensive when the active portion of your application will not fit into on-chip cache. When you are working with data on a large scale, it is still important to select the data type that can be processed most efficiently.

For example, integer data is processed faster than floating-point data. Floating-point data is defined according to internal database-specific formats, usually in a compressed format. The data must be decompressed and converted into a different format so that it can be processed by the wire protocol.

Processing time is shortest for character strings, followed by integers, which usually require some conversion or byte ordering. Processing floating-point data and timestamps is at least twice as slow as integers.

DB2 UDB on z/OS

The following table lists the data types that the ODBC driver supports for DB2 UDB on z/OS:

<i>Table B-1. Mapping Data Types for DB2 UDB on z/OS to ODBC Data Types</i>	
DB2 UDB Data Type	ODBC Data Type (ANSI)
BIGINT	SQL_BIGINT
BLOB	SQL_LONGVARBINARY
CHAR(size)	SQL_CHAR
CHAR(size) FOR BIT DATA	SQL_BINARY
CLOB	SQL_LONGVARCHAR
DATE	SQL_DATE
DBCLOB	SQL_WLONGVARCHAR
DECIMAL	SQL_DECIMAL
FLOAT	SQL_FLOAT
	SQL_DOUBLE
INTEGER	SQL_INTEGER
LONGVARCHAR	SQL_LONGVARCHAR
LONGVARBINARY	SQL_LONGVARBINARY
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLINT	SQL_SMALLINT
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP ¹
	SQL_CHAR ²
VARCHAR(size)	SQL_VARCHAR
VARCHAR(size) FOR BIT DATA	SQL_VARBINARY

1. Only present when DataSourceFetchTimestampAsString=False.
2. Only present when DataSourceFetchTimestampAsString=True.

The following table lists the data types that the ADO data provider supports for DB2 UDB for z/OS:

Table B-2. Mapping Data Types for DB2 UDB for z/OS to ADO Data Types

DB2 UDB Data Type	OLE DB Data Type
BIGINT	DBTYPE_I8
BLOB	DBTYPE_BYTES
CHAR(size)	DBTYPE_STR
CHAR(size) FOR BIT DATA	DBTYPE_BYTES
CLOB	DBTYPE_STR
DATE	DBTYPE_DBDATE
DECIMAL	DBTYPE_DECIMAL
FLOAT	DBTYPE_R8
INTEGER	DBTYPE_I4
LONG VARCHAR	DBTYPE_STR
LONG VARCHAR FOR BIT DATA	DBTYPE_BYTES
NUMERIC	DBTYPE_NUMERIC
REAL	DBTYPE_R4
SMALLINT	DBTYPE_I2
TIME	DBTYPE_DBTIME
TIMESTAMP	DBTYPE_DBTIMESTAMP ¹ DBTYPE_STR ²
VARCHAR(size)	DBTYPE_STR
VARCHAR(size) FOR BIT DATA	DBTYPE_BYTES

1. Only present when DataSourceFetchTimestampAsString=False.

2. Only present when DataSourceFetchTimestampAsString=True.

The following table lists the data types that the JDBC driver supports for DB2 UDB for z/OS:

Table B-3. Mapping Data Types for DB2 UDB on z/OS to JDBC Data Types	
DB2 Data Type	JDBC Data Type
BIGINT	BIGINT
BLOB	BLOB ¹
	LONGVARBINARY ²
CHAR(size)	CHAR
CHAR(size) FOR BIT DATA	BINARY
CLOB	CLOB ¹
	LONGVARCHAR ²
DATE	DATE
DBCLOB	CLOB
DECIMAL	DECIMAL
FLOAT	FLOAT
	DOUBLE
INTEGER	INTEGER
LONG VARCHAR	LONGVARCHAR ¹
LONG VARCHAR FOR BIT DATA	LONGVARBINARY ²
NUMERIC	NUMERIC
REAL	REAL
SMALLINT	SMALLINT
TIME	TIME
TIMESTAMP	TIMESTAMP ³
	CHAR ⁴
VARCHAR(size)	VARCHAR
VARCHAR(size) FOR BIT DATA	VARBINARY
<div><div>1. Only present when DataSourceReportLobsAsLongvar=False.</div><div>2. Only present when DataSourceReportLobsAsLongvar=True.</div><div>3. Only present when DataSourceFetchTimestampAsString=False.</div><div>4. Only present when DataSourceFetchTimestampAsString=True.</div></div>	

The following table maps the data types that the .NET data provider supports for DB2 UDB for z/OS:

Table B-4. Mapping Data Types for DB2 UDB on z/OS to .NET Data Types

DB2 UDB Data Type	ADO.NET Data Type
BIGINT	BIGINT
BLOB	BLOB
CHAR(size)	CHAR
CHAR(size) FOR BIT DATA	BINARY
CLOB	CLOB
DATE	DATE
DBCLOB	CLOB
DECIMAL	DECIMAL
FLOAT	DOUBLE
INTEGER	INT
LONG VARCHAR	LVARCHAR
LONG VARCHAR FOR BIT DATA	LVARBINARY
NUMERIC	DECIMAL
REAL	SINGLE
SMALLINT	SMALLINT
TIME	TIME
TIMESTAMP	TIMESTAMP ¹ VARCHAR ²
UNIQUEIDENTIFIER	
VARCHAR(size)	VARCHAR
VARCHAR(size) FOR BIT DATA	VARBINARY

1. Only present when DataSourceFetchTimestampAsString=False.

2. Only present when DataSourceFetchTimestampAsString=True.

DB2 UDB on Linux, UNIX, and Windows



 The following table lists the data types that the ODBC driver supports for DB2 UDB on Linux, UNIX, and Windows for the default service (ServiceCodePage=Default or OS) and the Unicode service (ServiceCodePage=Database):

Table B-5. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ODBC Data Types

DB2 SQL Data Type	ODBC Data Types (Non-Unicode Service)	ODBC Data Types (Unicode Service)
BIGINT	SQL_BIGINT	SQL_BIGINT
BLOB ¹	SQL_LONGVARBINARY	SQL_LONGVARBINARY
CHAR(size)	SQL_CHAR	SQL_CHAR ² SQL_WCHAR
CHAR(size) FOR BIT DATA	SQL_BINARY	SQL_BINARY
CLOB ¹	SQL_LONGVARCHAR	SQL_WLONGVARCHAR ³
DATE	SQL_DATE	SQL_DATE
DBCLOB ⁴	Not supported	SQL_WLONGVARCHAR
DECIMAL	SQL_DECIMAL	SQL_DECIMAL
FLOAT	SQL_DOUBLE SQL_FLOAT	SQL_DOUBLE SQL_FLOAT
GRAPHIC(size) ⁴	Not supported	SQL_WCHAR
INTEGER	SQL_INTEGER	SQL_INTEGER
LONG VARCHAR ¹	SQL_LONGVARCHAR	SQL_LONGVARCHAR ²
LONG VARCHAR FOR BIT DATA ¹	SQL_LONGVARBINARY	SQL_LONGVARBINARY
LONG VARGRAPHIC ⁴	Not supported	SQL_WLONGVARCHAR
NUMERIC	SQL_NUMERIC	SQL_NUMERIC
REAL	SQL_REAL	SQL_REAL
SMALLINT	SQL_SMALLINT	SQL_SMALLINT
TIME	SQL_TIME	SQL_TIME

Table B-5. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ODBC Data Types (cont.)

DB2 SQL Data Type	ODBC Data Types (Non-Unicode Service)	ODBC Data Types (Unicode Service)
TIMESTAMP	SQL_TIMESTAMP SQL_CHAR ⁵	SQL_TIMESTAMP SQL_CHAR ⁵
VARCHAR(size)	SQL_VARCHAR	SQL_VARCHAR ² SQL_WVARCHAR
VARCHAR(size) FOR BIT DATA	SQL_VARBINARY	SQL_VARBINARY
VARGRAPHIC(size) ⁴	Not supported	SQL_WVARCHAR

1. Depends on setting of the DataSourceLobAsLongvar service attribute. When set to False, LOB data types will be handled as LOB objects; when set to True, LOB data types are handled as Longvar data types.
2. If DataSourceReportChar is enabled, the CHAR, VARCHAR, LONGVARCHAR data types are additionally reported as the non-Unicode data types (see previous table)
3. Depending on DataSourceReportLobAsLongVarchar: if set to TRUE: LONGVARCHAR data types are reported; if set to FALSE: LOB data types are reported.
4. If DataSourceDB2GraphicSupport is enabled, the GRAPHIC datatypes will be available.
5. DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings.

The following table maps the data types that the ADO data provider supports for DB2 UDB. The ADO data provider does not support Unicode service for DB2 UDB.

Table B-6. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ADO Data Types

DB2 SQL Data Type	OLE DB Data Type Non-Unicode Service
BIGINT	DBTYPE_I8
BLOB ¹	DBTYPE_BYTES
CHAR	DBTYPE_STR

Table B-6. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to ADO Data Types *(cont.)*

DB2 SQL Data Type	OLE DB Data Type Non-Unicode Service
CHAR() FOR BIT DATA	DBTYPE_BYTES
CLOB ¹	DBTYPE_STR
DATE	DBTYPE_DBDATE
DECIMAL	DBTYPE_DECIMAL
FLOAT	DBTYPE_R8
INTEGER	DBTYPE_I4
LONG VARCHAR ¹	DBTYPE_STR
LONG VARCHAR FOR BIT DATA ¹	DBTYPE_BYTES
NUMERIC	DBTYPE_NUMERIC
REAL	DBTYPE_R4
SMALLINT	DBTYPE_I2
TIME	DBTYPE_DBTIME
TIMESTAMP	DBTYPE_DBTIMESTAMP
	DBTYPE_STR ²
VARCHAR(size)	DBTYPE_STR
VARCHAR(size) FOR BIT DATA	DBTYPE_BYTES

- 1. Depends on setting of the DataSourceLobAsLongvar service attribute. When set to False, LOB data types will be handled as LOB objects; when set to True, LOB data types are handled as Longvar data types.
 - 2. DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings.
-

The following table lists the SQL data types that the JDBC driver supports for DB2 UDB for the default service (ServiceCodePage=Default or OS) and the Unicode service (ServiceCodePage=Database):

Table B-7. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to JDBC Data Types

DB2 SQL Data Type	JDBC Data Type Non-Unicode Service	JDBC Data Type Unicode Service
BIGINT	BIGINT	BIGINT
BLOB	BLOB ¹	BLOB ²
	LONGVARBINARY ¹	LONGVARBINARY ²
CHAR(size)	CHAR	CHAR
CHAR(size) FOR BIT DATA	BINARY	BINARY
CLOB	CLOB ¹	CLOB ²
	LONGVARCHAR ¹	LONGVARCHAR ²
DATE	DATE	DATE
DBCLOB	CLOB ¹	CLOB ²
	LONGVARCHAR ¹	LONGVARCHAR ²
DECIMAL	DECIMAL	DECIMAL
FLOAT	FLOAT	FLOAT
	DOUBLE	DOUBLE
GRAPHIC(size)	CHAR	CHAR ³
INTEGER	INTEGER	INTEGER
LONG VARCHAR	LONGVARCHAR ¹	LONGVARCHAR ¹
LONG VARCHAR FOR BIT DATA	LONGVARBINARY ¹	LONGVARBINARY ¹
LONG VARGRAPHIC	LONGVARCHAR	LONGVARCHAR ³
NUMERIC	NUMERIC	NUMERIC
REAL	REAL	REAL
SMALLINT	SMALLINT	SMALLINT
TIME	TIME	TIME

Table B-7. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to JDBC Data Types (cont.)

DB2 SQL Data Type	JDBC Data Type Non-Unicode Service	JDBC Data Type Unicode Service
TIMESTAMP	TIMESTAMP CHAR ⁴	TIMESTAMP CHAR ⁴
VARCHAR(size)	VARCHAR	VARCHAR
VARCHAR(size) FOR BIT DATA	VARBINARY	VARBINARY
VARGRAPHIC(size)	VARCHAR	VARCHAR ⁵

1. Depends on setting of the DataSourceLobAsLongvar service attribute. When set to False, LOB data types will be handled as LOB objects; when set to True, LOB data types are handled as Longvar data types.

2. DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings.

3. If DataSourceReportChar is enabled, the CHAR, VARCHAR, LONGVARCHAR data types are additionally reported as the non-Unicode data types (see previous table).

4. If DataSourceDB2GraphicSupport is enabled, the GRAPHIC datatypes will be available.

5. Depending on DataSourceReportLobAsLongVarchar: if set to TRUE, LONGVARCHAR data types are reported; if set to FALSE, LOB data types are reported.

The following table lists the data types that the .NET data provider supports for DB2 UDB for the default service (ServiceCodePage=Default or OS) and the Unicode service (ServiceCodePage=Database).

Table B-8. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to .NET Data Types

DB2 Data Type	.NET Data Type Non-Unicode Service	.NET Data Type Unicode Service
BIGINT	Int64	Int64
BLOB	byte[]	byte[]
CHAR(size)	String	String
CHAR(size) FOR BIT DATA	byte[]	byte[]

Table B-8. Mapping Data Types for DB2 UDB on Linux/UNIX/Windows to .NET Data Types (cont.)

DB2 Data Type	.NET Data Type Non-Unicode Service	.NET Data Type Unicode Service
CLOB	String	String
DATE	DateTime	DateTime
DBCLOB	String	String
DECIMAL	Decimal	Decimal
FLOAT	Double	Double
GRAPHIC(size)	String	String
INTEGER	Int32	Int32
LONG VARCHAR	String	String
LONG VARCHAR FOR BIT DATA	byte[]	byte[]
LONG VARGRAPHIC(size)	String	String
NUMERIC	Decimal	Decimal
REAL	Single	Single
ROWID	Single	Single
SMALLINT	Int16	Int16
TIME	DateTime	DateTime
TIMESTAMP	DateTime	DateTime
	String ¹	String ¹
VARCHAR(size)	String	String
VARCHAR(size) FOR BIT DATA	byte[]	byte[]
VARGRAPHIC	String	String

1. DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings

Informix

The following table lists the data types that the ODBC driver supports for Informix:

<i>Table B-9. Mapping Informix Data Types to ODBC Data Types</i>	
Informix Data Type	ODBC Data Type
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
BYTE	SQL_LONGVARBINARY
CHAR	SQL_CHAR
CLOB	SQL_LONGVARCHAR
DATE	SQL__TYPE_DATE
DATETIME HOUR TO SECOND	SQL__TYPE_TIME
DATETIME YEAR TO FRACTION(5)	SQL__TYPE_TIMESTAMP
DECIMAL	SQL_DECIMAL
FLOAT	SQL_FLOAT
INTEGER	SQL_INTEGER
INT8	SQL_BIGINT
LVARCHAR	SQL_LONGVARCHAR
MONEY	SQL_DECIMAL
NUMERIC	SQL_NUMERIC
SERIAL	SQL_INTEGER
SERIAL8	SQL_BIGINT
SMALLFLOAT	SQL_REAL
SMALLINT	SQL_SMALLINT
TEXT	SQL_LONGVARCHAR
VARCHAR	SQL_VARCHAR

The following table lists the data types that the ADO data provider supports for Informix:

Table B-10. Mapping the Informix Data Types to ADO Data Types

Informix Data Type	OLE DB Data Type
BLOB	DBTYPE_BYTES
BOOLEAN	DBTYPE_BOOL
BYTE	DBTYPE_BYTES
CHAR	DBTYPE_STR
CLOB	DBTYPE_STR
DATE	DBTYPE_DBDATE
DATETIME HOUR TO SECOND	DBTYPE_DBTIME
DATETIME YEAR TO FRACTION(5)	DBTYPE_DBTIMESTAMP
DECIMAL	DBTYPE_NUMERIC
FLOAT	DBTYPE_R8
INTEGER	DBTYPE_I4
INT8	DBTYPE_I8
LVARCHAR	DBTYPE_STR
MONEY	DBTYPE_NUMERIC
NUMERIC	DBTYPE_NUMERIC
SERIAL	DBTYPE_I4
SERIAL8	DBTYPE_I8
SMALLFLOAT	DBTYPE_R4
SMALLINT	DBTYPE_I2
TEXT	DBTYPE_STR
VARCHAR	DBTYPE_STR

The following table lists the data types that the JDBC driver supports for Informix:

<i>Table B-11. Mapping Informix Data Types to JDBC Data Types</i>	
Informix Data Type	JDBC Data Type
BLOB ¹	LONGVARBINARY ²
BOOLEAN	BIT
BYTE	LONGVARBINARY
CHAR	CHAR
CLOB ¹	LONGVARCHAR ²
DATE	DATE
DATETIME HOUR TO SECOND	TIME
DATETIME YEAR TO FRACTION(5)	TIMESTAMP
DECIMAL	DECIMAL
FLOAT	FLOAT
INTEGER	INTEGER
INT8	BIGINT
LVARCHAR	LONGVARCHAR
MONEY	DECIMAL
NUMERIC	NUMERIC
SERIAL	INTEGER
SERIAL8	BIGINT
SMALLFLOAT	REAL
SMALLINT	SMALLINT
TEXT	LONGVARCHAR
VARCHAR	VARCHAR
1. Only present when DataSourceReportLobsAsLongvar=False	
2. Only present when DataSourceReportLobsAsLongvar=True	

The following table lists the data types that the .NET data provider supports for Informix:

Table B-12. Mapping Informix Data Types to .NET Data Types

Informix Data Type	.NET Data Type
BLOB	Byte[]
BOOLEAN	Boolean
BYTE	Byte[]
CHAR	String
CLOB	Byte[]
DATE	DateTime
DATETIME HOUR TO SECOND	DateTime
DATETIME YEAR TO FRACTION(5)	DateTime
DECIMAL	Decimal
FLOAT	Double
INTEGER	Int32
INT8	Int64
LVARCHAR	String
MONEY	Decimal
NUMERIC	Decimal
SERIAL	Int32
SERIAL8	Int64
SMALLFLOAT	Float
SMALLINT	Int16
TEXT	String
VARCHAR	String

Microsoft SQL Server

The following table maps the ODBC data types that the ODBC driver supports to the Microsoft SQL Server:

Table B-13. Mapping Data Types for Microsoft SQL Server to ODBC Data Types

Microsoft SQL Server 2000, 2005 Data Type	ODBC Data Type
bigint	SQL_BIGINT
bigint identity	SQL_BIGINT
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR ¹
	SQL_WCHAR ²
datetime	SQL_DATETIME ³
	SQL_CHAR ⁴
decimal	SQL_DECIMAL
decimal identity	SQL_DECIMAL
float	SQL_FLOAT
guid	SQL_BINARY
image	SQL_LONGVARBINARY
int	SQL_INTEGER
int identity	SQL_INTEGER
money	SQL_DECIMAL
nchar	SQL_CHAR ¹
	SQL_WCHAR ²
ntext	SQL_LONGVARCHAR ¹
	SQL_WLONGVARCHAR ²
nvarchar(max)	SQL_LONGVARCHAR
numeric	SQL_NUMERIC
numeric identity	SQL_NUMERIC

Table B-13. Mapping Data Types for Microsoft SQL Server to ODBC Data Types (cont.)

Microsoft SQL Server 2000, 2005

Data Type	ODBC Data Type
nvarchar	SQL_VARCHAR ¹ SQL_WVARCHAR ²
real	SQL_REAL
smalldatetime	SQL_DATETIME ³ SQL_CHAR ⁴
smallint	SQL_SMALLINT
smallint identity	SQL_SMALLINT
smallmoney	SQL_DECIMAL
SQLVariant	SQL_BINARY
sysname	SQL_VARCHAR ¹ SQL_WVARCHAR ²
text	SQL_LONGVARCHAR ¹ SQL_WVARCHAR ²
timestamp	SQL_BINARY
tinyint	SQL_TINYINT
tinyint identity	SQL_TINYINT
uniqueidentifier	SQL_BINARY
varbinary	SQL_VARBINARY
varbinary(max)	SQL_LONGVARBINARY
varchar	SQL_VARCHAR ¹ SQL_WVARCHAR ²
varchar(max)	SQL_LONGVARCHAR ¹

1. Not present when DataSourceReportChar=False and ServiceCodePage=Database
 2. Only present whe ServiceCodePage=Database
 3. Only present when DataSourceFetchTimestampAsString=False
 4. Only present when DataSourceFetchTimestampAsString=True
-

The following table lists the data types that the ADO data provider supports for Microsoft SQL Server:

Table B-14. Mapping Microsoft SQL Server Data Types to ADO Data Types

Microsoft SQL Server 2000, 2005 Data Type	OLE DB Data Type
bigint	DBTYPE_I8
bigint identity	DBTYPE_I8
binary	DBTYPE_BYTES
bit	DBTYPE_BOOL
char	DBTYPE_STR
datetime	DBTYPE_DBTIMESTAMP ¹ DBTYPE_STR ²
decimal	DBTYPE_NUMERIC
decimal identity	DBTYPE_NUMERIC
float	DBTYPE_R8
guid	DBTYPE_GUID
image	DBTYPE_BYTES
int	DBTYPE_I4
int identity	DBTYPE_I4
money	DBTYPE_NUMERIC
nchar	DBTYPE_STR ³
nvarchar(max)	DBTYPE_STR
ntext	DBTYPE_STR ³
numeric	DBTYPE_NUMERIC
numeric() identity	DBTYPE_NUMERIC
nvarchar	DBTYPE_STR ³
real	DBTYPE_R4
smalldatetime	DBTYPE_DBTIMESTAMP ¹ DBTYPE_STR ²
Smallint	DBTYPE_I2

Table B-14. Mapping Microsoft SQL Server Data Types to ADO Data Types (cont.)

Microsoft SQL Server 2000, 2005 Data Type	OLE DB Data Type
Smallint identity	DBTYPE_I2
Smallmoney	DBTYPE_NUMERIC
sqlvariant	DBTYPE_BYTES
sysname	DBTYPE_STR ³
text	DBTYPE_STR
timestamp	DBTYPE_BYTES
tinyint	DBTYPE_UI1
tinyint identity	DBTYPE_I1
uniqueidentifier	DBTYPE_GUID
varbinary	DBTYPE_BYTES
varchar	DBTYPE_STR

1. Limited nchar,nvarchar, ntext support only
2. Only present when DataSourceFetchTimestampAsString=False
3. Only present when DataSourceFetchTimestampAsString=True

The following table lists the data types that the JDBC driver supports for Microsoft SQL Server:

Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types

Microsoft SQL Server 2000, 2005 Data Type	JDBC Data Type
bigint	BIGINT
bigint identity	BIGINT
binary	BINARY
bit	BIT
char	CHAR

Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types *(cont.)*

Microsoft SQL Server 2000, 2005	
Data Type	JDBC Data Type
datetime	TIMESTAMP ¹
	CHAR ²
decimal	DECIMAL
decimal identity	DECIMAL
float	FLOAT
guid	BINARY
image	LONGVARBINARY
int	INTEGER
int identity	INTEGER
money	DECIMAL
nchar	CHAR ³
ntext	LONGVARCHAR ³
nvarchar(max)	LONGVARCHAR ³
numeric	NUMERIC
numeric identity	NUMERIC
nvarchar	VARCHAR ³
real	REAL
smalldatetime	TIMESTAMP ¹
	CHAR ²
smallint	SMALLINT
smallint identity	SMALLINT
smallmoney	DECIMAL
sqlvariant	BINARY
sysname	VARCHAR ³
text	LONGVARCHAR
timestamp	BINARY
tinyint	TINYINT
tinyint identity	TINYINT

Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types (cont.)

Microsoft SQL Server 2000, 2005

Data Type	JDBC Data Type
uniqueidentifier	BINARY
varbinary	VARBINARY
varchar	VARCHAR

1. Only present when DataSourceFetchTimestampAsString=False
 2. Only present when DataSourceFetchTimestampAsString=False
 3. Full nchar, nvarchar, and ntext support requires ServiceCodePage=Database
-

The following table lists the data types that the .NET data provider supports for Microsoft SQL Server:

Table B-16. Mapping Microsoft SQL Server Data Types to .NET Framework Types

Microsoft SQL Server 2000, 2005

Data Type	.NET Framework Type
bigint	Int64
bigint identity	Int64
binary	Byte[]
bit	Boolean
char	String
	Char[]
datetime	DateTime
decimal	Decimal
decimal identity	Decimal
float	Double
guid	Byte[]
image	Byte[]

Table B-16. Mapping Microsoft SQL Server Data Types to .NET Framework Types *(cont.)*

Microsoft SQL Server 2000, 2005 Data Type	.NET Framework Type
int	Int32
int identity	Int32
money	Decimal
nchar	String
	Char[]
ntext	String
	Char[]
numeric	Decimal
numeric() identity	Decimal
nvarchar ¹	String
	Char[]
nvarchar(max)	String
	Char[]
real	Single
smalldatetime	DateTime
smallint	Int16
smallint identity	INT
smallmoney	Decimal
SQLVariant	Bytes[]
sysname	Byte[]
text	String
	Char[]
timestamp	Byte[]
tinyint	Byte
tinyint identity	Byte[]
uniqueidentifier	Byte[]
varbinary	Byte[]
varbinary(max)	Byte[]

Table B-16. Mapping Microsoft SQL Server Data Types to .NET Framework Types (cont.)

Microsoft SQL Server 2000, 2005 Data Type	.NET Framework Type
varchar	String Char[]
varchar(max)	String Char[]

Oracle

The following table lists the data types that the ODBC driver supports for Oracle:

Table B-17. Mapping the Oracle Data Types to ODBC Data Types

Oracle 9i Data Type	Oracle10g Data Type	ODBC Data Type
BFILE	BFILE	SQL_LONGVARBINARY
	BINARY_FLOAT	SQL_REAL
	BINARY_DOUBLE	SQL_DOUBLE
BLOB	BLOB	SQL_LONGVARBINARY
CHAR(size)	CHAR(size)	SQL_CHAR ¹ SQL_WCHAR ²
CLOB	CLOB	SQL_LONGVARCHAR ³ SQL_WLONGVARCHAR ²
DATE	DATE	SQL_TYPE_TIMESTAMP

Table B-17. Mapping the Oracle Data Types to ODBC Data Types
(cont.)

Oracle 9i Data Type	Oracle10g Data Type	ODBC Data Type
LONG	LONG	SQL_LONGVARCHAR ¹ SQL_WLONGVARCHAR ²
LONG RAW	LONG RAW	SQL_LONGVARBINARY
NCHAR(size)	NCHAR(size)	SQL_CHAR ⁴ SQL_WCHAR ⁵
NCLOB	NCLOB	SQL_LONGVARCHAR ⁴ SQL_WLONGVARCHAR ⁵
NUMBER	NUMBER	SQL_FLOAT
NUMBER(p,s)	NUMBER(p,s)	SQL_DECIMAL
NVARCHAR2(size)	NVARCHAR2(size)	SQL_VARCHAR ⁴ SQL_WVARCHAR ⁵
RAW(size)	RAW(size)	SQL_VARBINARY
ROWID	ROWID	SQL_VARCHAR
TIMESTAMP	TIMESTAMP	SQL_TIMESTAMP ⁶ SQL_CHAR ⁷
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP WITH LOCAL TIME ZONE	SQL_TYPE_TIMESTAMP ⁶ SQL_CHAR ⁷
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE	SQL_TYPE_TIMESTAMP ^{6, 8, 9}
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE	SQL_CHAR ^{7, 8}
VARCHAR2(size)		SQL_VARCHAR ¹
VARCHAR2(size)		SQL_WVARCHAR ²

- 1. Not present when DataSourceReportChar=False, ServiceCodePage= Database and database character set is Unicode
- 2. Only present when ServiceCodePage=Database and database character set is Unicode
- 3. Not present when ServiceCodePage=Database and database character set is Unicode

4. Not present when ServiceCodePage=Database and database character set is Unicode
 5. Only present when ServiceCodePage=Database and national character set is Unicode
 6. Only present when DataSourceFetchTimestampAsString=False
 7. Only present when DataSourceFetchTimestampAsString=True
 8. Only present when DataSourceORAMapTSWTZ=True
 9. The time zone value is not accessible (or can not be set). Instead, the client time zone is used either to pass to Oracle or to convert from the stored time zone value.
-

The following table lists the data types that the ADO data provider supports for Oracle:

Table B-18. Mapping the Data Types for Oracle to ADO Data Types

Oracle 9i, 10g Data Type	OLE DB Data Type
BFILE	DBTYPE_BYTES
BLOB	DBTYPE_BYTES
CHAR(size)	DBTYPE_STR
CLOB	DBTYPE_STR
DATE	DBTYPE_DBTIMESTAMP
LONG	DBTYPE_STR
LONG RAW	DBTYPE_BYTES
NCHAR(size)	DBTYPE_STR ¹
NCLOB	DBTYPE_STR ¹
NUMBER	DBTYPE_R8
NUMBER(p,s)	DBTYPE_NUMERIC
NVARCHAR2(size)	DBTYPE_STR ¹
RAW(size)	DBTYPE_BYTES
ROWID	DBTYPE_STR
TIMESTAMP	DBTYPE_DBTIMESTAMP ²
	DBTYPE_STR ³

Table B-18. Mapping the Data Types for Oracle to ADO Data Types (cont.)

Oracle 9i, 10g Data Type	OLE DB Data Type
TIMESTAMP WITH LOCAL TIME ZONE	DBTYPE_DBTIMESTAMP ² DBTYPE_STR ³
TIMESTAMP WITH TIME ZONE	DBTYPE_DBTIMESTAMP ^{2, 4, 5} DBTYPE_STR ^{3, 4}
VARCHAR2	DBTYPE_STR

1. The Unicode data types Nchar, Nclob, and Nvarchar2 are partially supported.

2. Only present when DataSourceFetchTimestampAsString=False

3. Only present when DataSourceFetchTimestampAsString=True

4. Only present when DataSourceORAMapTSWTZ=True

5. The time zone value is not accessible (or can not be set). Instead, the client time zone is used either to pass to Oracle or to convert from the stored time zone value.

The following table lists the data types that the JDBC driver supports for Oracle:

Table B-19. Mapping the Data Types for Oracle to JDBC Data Types

Oracle9i Data Type	Oracle10g Data Type	JDBC Data Type
BFILE	BFILE	LONGVARBINARY
	BINARY_FLOAT	SQL_REAL
	BINARYDOUBLE	SQL_DOUBLE
BLOB	BLOB	BLOB ¹ LONGVARBINARY ²
CHAR(size)	CHAR(size)	CHAR
CLOB	CLOB	CLOB ¹ LONGVARCHAR ²

Table B-19. Mapping the Data Types for Oracle to JDBC Data Types (cont.)

Oracle9i Data Type	Oracle10g Data Type	JDBC Data Type
DATE	DATE	TIMESTAMP
LONG	LONG	LONGVARCHAR ¹
LONG RAW	LONG RAW	LONGVARBINARY ¹
NCHAR(size)	NCHAR(size)	CHAR
NCLOB	NCLOB	CLOB ¹
		LONGVARCHAR ²
NUMBER	NUMBER	FLOAT
NUMBER(p,s)	NUMBER(p,s)	DECIMAL
NVARCHAR2(size)	NVARCHAR2(size)	VARCHAR
RAW(size)	RAW(size)	VARBINARY
ROWID	ROWID	VARCHAR
VARCHAR2(size)	VARCHAR2(size)	VARCHAR
TIMESTAMP	TIMESTAMP	TIMESTAMP ³
		CHAR ⁴
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP ³
		CHAR ⁴
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE	TIMESTAMP ^{3, 5, 6}
		CHAR ^{4, 5}

1. Only present when DataSourceReportLobsAsLongvar=False
 2. Only present when DataSourceReportLobsAsLongvar=True
 3. Only present when DataSourceFetchTimestampAsString=False
 4. Only present when DataSourceFetchTimestampAsString=True
 5. Only present when DataSourceORAMapTSWTZ=True
 6. The time zone value is not accessible or cannot be set. Instead, the client time zone is used to pass to Oracle or to convert from the stored time zone value.
-

The following table lists the data types that the .NET data provider supports for Oracle:

Table B-20. Mapping Oracle Data Types to .NET Framework Types		
Oracle9i Data Type	Oracle10g Data Type	.NET Framework Type
BFILE		Byte[]
	BINARY_DOUBLE	Decimal
	BINARY_FLOAT	Decimal
BLOB	BLOB	Byte[]
CHAR(size)	CHAR(size)	String
		Char[]
CLOB	CLOB	String
DATE	DATE	DateTime
LONG	LONG	String
		Char[]
LONG RAW	LONG RAW	Byte[]
NCHAR(size)	NCHAR(size)	String
		Char[]
NCLOB	NCLOB	String
		Char[]
NUMBER	NUMBER	Decimal
NUMBER(p,s)	NUMBER(p,s)	Decimal
NVARCHAR2(size)	NVARCHAR2(size)	String
		Char[]
RAW(size)	RAW(size)	Byte[]
ROWID	ROWID	String
		Char[]
TIMESTAMP	TIMESTAMP	DateTime
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP WITH LOCAL TIME ZONE	DateTime

Table B-20. Mapping Oracle Data Types to .NET Framework Types *(cont.)*

Oracle9i Data Type	Oracle10g Data Type	.NET Framework Type
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE	DateTime
VARCHAR2(size)	VARCHAR2(size)	String Char[]

Sybase

The following table lists the data types that the ODBC driver supports for Sybase:

Table B-21. Mapping the Data Types for Sybase to ODBC Data Types

Sybase Data Type	ODBC Data Type
BIGINT	SQL_BIGINT
BINARY	SQL_BINARY
BIT	SQL_BIT
CHAR	SQL_CHAR ¹
	SQL_WCHAR ²
DATETIME	SQL_TIMESTAMP ³
	SQL_CHAR ⁴
DECIMAL	SQL_DECIMAL
FLOAT	SQL_FLOAT
IMAGE	SQL_LONGVARBINARY
INT	SQL_INTEGER
MONEY	SQL_DECIMAL
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLDATETIME	SQL_TIMESTAMP ³
	SQL_CHAR ⁴
SMALLINT	SQL_SMALLINT
SMALLMONEY	SQL_DECIMAL
SYSNAME	SQL_VARCHAR ⁵
	SQL_WVARCHAR ²
TEXT	SQL_LONGVARCHAR ¹
	SQL_WLONGVARCHAR ²
TIMESTAMP	SQL_BINARY
TINYINT	SQL_TINYINT

Table B-21. Mapping the Data Types for Sybase to ODBC Data Types (cont.)

Sybase Data Type	ODBC Data Type
UBIGINT	SQL_DECIMAL
UNICHAR	SQL_WCHAR ⁶
UNIVARCHAR	SQL_WVARCHAR ⁶
VARBINARY	SQL_VARBINARY
VARCHAR	SQL_VARCHAR ¹
	SQL_WVARCHAR ²

1. Not present when DataSourceReportChar=False and ServiceCodePage=Database
 2. Only present when ServiceCodePage=Database
 3. Only present when DataSourceFetchTimestampAsString=False
 4. Only present when DataSourceFetchTimestampAsString=True
 5. Not present when ServiceCodePage=Database
 6. Only present when ServiceCodePage=Database and database character set is UTF-8
-

The following table lists the data types that the ADO data provider supports for Sybase:

Table B-22. Mapping the Data Types for Sybase to ADO Data Types

Sybase Data Type	OLE DB Data Type
BIGINT	DBTYPE_I8
BINARY	DBTYPE_BYTES
BIT	DBTYPE_BOOL
CHAR	DBTYPE_STR
DATETIME	DBTYPE_DBTIMESTAMP ¹
	DBTYPE_STR ²
DECIMAL	DBTYPE_NUMERIC
FLOAT	DBTYPE_R8

Table B-22. Mapping the Data Types for Sybase to ADO Data Types *(cont.)*

Sybase Data Type	OLE DB Data Type
IMAGE	DBTYPE_BYTES
INT	DBTYPE_I4
MONEY	DBTYPE_CY
NUMERIC	DBTYPE_R4
REAL	DBTYPE_R4
SMALLDATETIME	DBTYPE_DBTIMESTAMP ¹
	DBTYPE_STR ²
SMALLINT	DBTYPE_I2
SMALLMONEY	DBTYPE_CY
SYSNAME	DBTYPE_STR
TEXT	DBTYPE_STR
TIMESTAMP	DBTYPE_BYTES
TINYINT	DBTYPE_UI1
UBIGINT	DBTYPE_I8
VARBINARY	DBTYPE_BYTES
VARCHAR	DBTYPE_STR

1. Only present when DataSourceFetchTimestampAsString=False
2. Only present when DataSourceFetchTimestampAsString=True

The following table lists the data types supported by the JDBC driver for Sybase:

Table B-23. Mapping the Data Types for Sybase to JDBC Data Types

Sybase Data Type	JDBC Data Type
BIGINT	BIGINT
BINARY	BINARY
BIT	BIT
CHAR	CHAR
DATETIME	TIMESTAMP ¹
	CHAR ²
DECIMAL	DECIMAL
FLOAT	FLOAT
IMAGE	LONGVARBINARY
INT	INTEGER
MONEY	DECIMAL
NUMERIC	NUMERIC
REAL	REAL
SMALLDATETIME	TIMESTAMP ¹
	CHAR ²
SMALLINT	SMALLINT
SMALLMONEY	DECIMAL
SYSNAME	VARCHAR
TEXT	LONGVARCHAR
TIMESTAMP	BINARY
TINYINT	TINYINT
UBIGINT	BIGINT
UNICHAR	CHAR ³

Table B-23. Mapping the Data Types for Sybase to JDBC Data Types *(cont.)*

Sybase Data Type	JDBC Data Type
UNIVARCHAR	VARCHAR ³
VARBINARY	VARBINARY
VARCHAR	VARCHAR

1. Only present when DataSourceFetchTimestampAsString=False

2. Only present when DataSourceFetchTimestampAsString=True

3. Only present when ServiceCodePage=Database and database character set is UTF-8

The following table lists the data types that the .NET data provider supports for Sybase:

Table B-24. Mapping Data Types for Sybase Data Types to .NET Framework Types

Sybase Data Type	.NET Framework Type
BIGINT	Byte[]
BINARY	Byte[]
BIT	Byte[]
CHAR	String
	Char[]
DATETIME	DateTime
DECIMAL	Decimal
FLOAT	Double
IMAGE	Byte[]
INT	Int32
MONEY	Decimal
NUMERIC	Decimal
REAL	Single

Table B-24. Mapping Data Types for Sybase Data Types to .NET Framework Types *(cont.)*

Sybase Data Type	.NET Framework Type
SMALLDATETIME	DateTime
SMALLINT	Int16
SMALLMONEY	Decimal
SYSNAME	String
TEXT	String
	Char[]
TIMESTAMP	Byte[]
TINYINT	Byte[]
UBIGINT	Int64
UNICHAR	Char
UNIVARCHAR	String
	Char[]
VARBINARY	Byte[]
VARCHAR	String
	Char[]

Isolation Levels

This section discusses the isolation levels supported by SequeLink.

[Table B-25](#) lists the isolation levels supported by each data store, including their default isolation level.

Table B-25. Isolation Levels

Database	Isolation Levels	Default
DB2 UDB on z/OS	Read uncommitted	Read committed
	Read committed	
	Repeatable read	
	Serializable	
DB2 UDB on Linux/UNIX/Windows	Read uncommitted	Read committed
	Read committed	
	Repeatable read	
	Serializable	
Informix	Read uncommitted	Read committed
	Read committed	
	Repeatable read	
Microsoft SQL Server	Read uncommitted	Read committed
	Read committed	
	Read Committed with Snapshots ¹	
	Read Committed with Locks ¹	
	Repeatable read	
	Serializable	
	Snapshot ¹	

Table B-25. Isolation Levels (cont.)

Database	Isolation Levels	Default
Oracle	Read committed Serializable	Read committed
Sybase	Read uncommitted Read committed Repeatable read Serializable	Read committed
1. Supported on Microsoft SQL Server only. See “Using Snapshot Isolation Level (Microsoft SQL Server 2005 Only)” on page 487 for more information.		

Using Snapshot Isolation Level (Microsoft SQL Server 2005 Only)

The Snapshot isolation level is available only with Microsoft SQL Server 2005. Enabling the SnapshotSerializable connection option changes the behavior of the Serializable isolation level to use the Snapshot Isolation level. This allows an application to use the Snapshot Isolation level with no or minimum code changes.

To configure Snapshot Isolation for connections, you must have your Microsoft SQL Server 2005 database configured for Snapshot Isolation and your application must have the transaction isolation level set to Serializable.

See the following sections for the connection information specific to the SequeLink Client.

Using The Snapshot Isolation Level with the ODBC Driver

If you are writing a new application, you may want to code it to set the connection attribute `SQL_COPT_SS_TXN_ISOLATION` to the value `SQL_TXN_SS_SNAPSHOT`. The application then uses the snapshot isolation level without requiring the Use Snapshot Transactions connection option.

C JDBC Support

This appendix provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.

JDBC Compatibility

SequeLink supports all JDKs from version 1.4.2 or higher, and the corresponding JDBC level.

Supported Functionality

The following tables list functionality supported for each JDBC object.

Array Object

Array Object Methods	Version Introduced	Supported	Comments
(all)	2.0 Core	No	Array objects are neither exposed, nor taken as input.

Blob Object

Blob Object Methods	Version Introduced	Supported	Comments
InputStream getBinaryStream ()	2.0 Core	Yes	
byte[] getBytes (long, int)	2.0 Core	Yes	
long length ()	2.0 Core	Yes	
long position (byte, long)	2.0 Core	Yes	
long position (Blob, long)	2.0 Core	Yes	
OutputStream setBinaryStream (long)	3.0 Core	Yes	
int setBytes (long, byte[])	3.0 Core	Yes	
int setBytes (long, byte[], int, int)	3.0 Core	Yes	
void truncate (long)	3.0 Core	Yes	
NOTE: Blob support is emulated using LONGVARBINARY datatypes on SQL Server, Sybase, JDBC Socket, and ODBC Socket. Using emulated Blobs incurs a performance and scalability penalty.			

CallableStatement Object

CallableStatement Object Methods	Version Introduced	Supported	Comments
Array getArray (int)	2.0 Core	No	Throws “unsupported method” exception.
Array getArray (String)	3.0	No	Throws “unsupported method” exception.

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
BigDecimal getBigDecimal (int)	2.0 Core	Yes	
BigDecimal getBigDecimal (int, int)	1.0	Yes	
BigDecimal getBigDecimal (String)	3.0	No	Throws “unsupported method” exception.
Blob getBlob (int)	2.0 Core	Yes	Not supported on servers which emulate this functionality over LONGVARBINARY.
Blob getBlob (String)	3.0	No	Throws “unsupported method” exception.
boolean execute ()	1.0	Yes	
boolean execute (String)	1.0	No	Throws “invalid method call” exception.
boolean execute (String, int)	3.0	No	Throws “invalid method call” exception.
boolean execute (String, int[])	3.0	No	Throws “invalid method call” exception.
boolean execute (String, String[])	3.0	No	Throws “invalid method call” exception.
boolean getBoolean (int)	1.0	Yes	
boolean getBoolean (String)	3.0	No	Throws “unsupported method” exception.
boolean getMoreResults ()	1.0	Yes	
boolean getMoreResults (int)	3.0	Yes	Not supported on Informix.
boolean wasNull ()	1.0	Yes	
byte [] getBytes (int)	1.0	Yes	

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
byte [] getBytes (String)	3.0	No	Throws “unsupported method” exception.
byte getByte (int)	1.0	Yes	
byte getByte (String)	3.0	No	Throws “unsupported method” exception.
Clob getClob (int)	2.0 Core	Yes	Not supported on servers which emulate this functionality over LONGVARCHAR.
Clob getClob (String)	3.0	No	Throws “unsupported method” exception.
Connection getConnection ()	2.0 Core	Yes	
Date getDate (int)	1.0	Yes	
Date getDate (int, Calendar)	2.0 Core	Yes	
Date getDate (String)	3.0	No	Throws “unsupported method” exception.
Date getDate (String, Calendar)	3.0	No	Throws “unsupported method” exception.
double getDouble (int)	1.0	Yes	
double getDouble (String)	3.0	No	Throws “unsupported method” exception.
float getFloat (int)	1.0	Yes	
float getFloat (String)	3.0	No	Throws “unsupported method” exception.
int [] executeBatch ()	2.0 Core	Yes	
int executeUpdate ()	1.0	Yes	
int executeUpdate (String)	1.0	No	Throws “invalid method call” exception.

CallableStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
int executeUpdate (String, int)	3.0	No	Throws “invalid method call” exception.
int executeUpdate (String, int[])	3.0	No	Throws “invalid method call” exception.
int executeUpdate (String, String[])	3.0	No	Throws “invalid method call” exception.
int getFetchDirection ()	2.0 Core	Yes	
int getFetchSize ()	2.0 Core	Yes	
int getInt (int)	1.0	Yes	
int getInt (String)	3.0	No	Throws “unsupported method” exception.
int getMaxFieldSize ()	1.0	Yes	
int getMaxRows ()	1.0	Yes	
int getQueryTimeout ()	1.0	Yes	Returns 0 for DB2 and Informix.
int getResultSetConcurrency ()	2.0 Core	Yes	
int getResultSetHoldability ()	3.0	No	Throws “unsupported method” exception.
int getResultSetType ()	2.0 Core	Yes	
int getUpdateCount ()	1.0	Yes	
long getLong (int)	1.0	Yes	
long getLong (String)	3.0	No	Throws “unsupported method” exception.
Object getObject (int)	1.0	Yes	
Object getObject (int, Map)	2.0 Core	Yes	Map ignored.
Object getObject (String)	3.0	No	Throws “unsupported method” exception.

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
Object getObject (String, Map)	3.0	No	Throws “unsupported method” exception.
ParameterMetaData getParameterMetaData()	3.0	Yes	Not supported on Oracle or Informix Servers.
Ref getRef (int)	2.0 Core	No	Throws “unsupported method” exception.
Ref getRef (String)	3.0	No	Throws “unsupported method” exception.
ResultSet executeQuery ()	1.0	Yes	
ResultSet executeQuery (String)	1.0	No	Throws “invalid method call” exception.
ResultSet getGeneratedKeys()	3.0	No	Throws “unsupported method” exception.
ResultSet getResultSet ()	1.0	Yes	
ResultSetMetaData getMetaData ()	2.0 Core	Yes	
short getShort (int)	1.0	Yes	
short getShort (String)	3.0	No	Throws “unsupported method” exception.
SQLWarning getWarnings ()	1.0	Yes	
String getString (int)	1.0	Yes	
String getString (String)	3.0	No	Throws “unsupported method” exception.
Time getTime (int)	1.0	Yes	
Time getTime (int, Calendar)	2.0 Core	Yes	
Time getTime (String)	3.0	No	Throws “unsupported method” exception.

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
Time getTime (String, Calendar)	3.0	No	Throws “unsupported method” exception.
Timestamp getTimestamp (int)	1.0	Yes	
Timestamp getTimestamp (int, Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (String)	3.0	No	Throws “unsupported method” exception.
Timestamp getTimestamp (String, Calendar)	3.0	No	Throws “unsupported method” exception.
URL getURL (int)	3.0	No	Throws “unsupported method” exception.
URL getURL (String)	3.0	No	Throws “unsupported method” exception.
void addBatch ()	2.0 Core	Yes	
void addBatch (String)	2.0 Core	No	Throws “invalid method call” exception.
void cancel ()	1.0	Yes	Cancel can be used to cancel a function running synchronously on a statement, using a different thread. Whether Cancel will actually cancel the running function depends on the data store.
void clearBatch ()	2.0 Core	Yes	
void clearParameters ()	1.0	Yes	
void clearWarnings ()	1.0	Yes	
void close ()	1.0	Yes	

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void registerOutParameter (int, int)	1.0	Yes	This method should only be used against data stores supporting output or input-output parameters.
void registerOutParameter (int, int, int)	1.0	Yes	This method should only be used against data stores supporting output or input-output parameters.
void registerOutParameter (int, int, String)	2.0 Core	Yes	String/typename ignored. This method should only be used against data stores supporting output or input-output parameters.
void registerOutParameter (String, int)	3.0	No	Throws “unsupported method” exception.
void registerOutParameter (String, int, int)	3.0	No	Throws “unsupported method” exception.
void registerOutParameter (String, int, String)	3.0	No	Throws “unsupported method” exception.
void setArray (int, Array)	2.0 Core	No	Throws “unsupported method” exception.
void setAsciiStream (int, InputStream, int)	1.0	Yes	
void setAsciiStream (String, InputStream, int)	3.0	No	Throws “unsupported method” exception.
void setBigDecimal (int, BigDecimal)	1.0	Yes	
void setBigDecimal (String, BigDecimal)	3.0	No	Throws “unsupported method” exception.
void setBinaryStream (int, InputStream, int)	1.0	Yes	

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void setBinaryStream (String, InputStream, int)	3.0	No	Throws "unsupported method" exception.
void setBlob (int, Blob)	2.0 Core	Yes	
void setBoolean (int, boolean)	1.0	Yes	
void setBoolean (String, boolean)	3.0	No	Throws "unsupported method" exception.
void setByte (int, byte)	1.0	Yes	
void setByte (String, byte)	3.0	No	Throws "unsupported method" exception.
void setBytes (int, byte [])	1.0	Yes	
void setBytes (String, byte [])	3.0	No	Throws "unsupported method" exception.
void setCharacterStream (int, Reader, int)	2.0 Core	Yes	
void setCharacterStream (String, Reader, int)	3.0	No	Throws "unsupported method" exception.
void setClob (int, Clob)	2.0 Core	Yes	
void setCursorName (String)	1.0	No	Throws "unsupported method" exception.
void setDate (int, Date)	1.0	Yes	SQL Server throws "optional feature not implemented" exception.
void setDate (int, Date, Calendar)	2.0 Core	Yes	SQL Server throws "optional feature not implemented" exception.
void setDate (String, Date)	3.0	No	Throws "unsupported method" exception.
void setDate (String, Date, Calendar)	3.0	No	Throws "unsupported method" exception.

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void setDouble (int, double)	1.0	Yes	
void setDouble (String, double)	3.0	No	Throws “unsupported method” exception.
void setEscapeProcessing (boolean)	1.0	Yes	
void setFetchDirection (int)	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.
void setFetchSize (int)	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.
void setFloat (int, float)	1.0	Yes	
void setFloat (String, float)	3.0	No	Throws “unsupported method” exception.
void setInt (int, int)	1.0	Yes	
void setInt (String, int)	3.0	No	Throws “unsupported method” exception.
void setLong (int, long)	1.0	Yes	SQL Server throws "optional feature not implemented" exception. However, you can use the workaround MSSMapLongToDecimal, which maps setLong to DECIMAL instead of BIGINT.
void setLong (String, long)	3.0	No	Throws “unsupported method” exception.
void setMaxFieldSize (int)	1.0	Yes	
void setMaxRows (int)	1.0	Yes	

CallableStatement			
Object <i>(cont.)</i>	Version		
Methods	Introduced	Supported	Comments
void setNull (int, int)	1.0	Yes	
void setNull (int, int, String)	2.0 Core	Yes	SequeLink does not support the Advanced Data Type functionality. This method is always mapped to setNull (i, sqlType).
void setNull (String, int)	3.0	No	Throws "unsupported method" exception.
void setNull (String, int, String)	3.0	No	Throws "unsupported method" exception.
void setObject (int, Object)	1.0	Yes	
void setObject (int, Object, int)	1.0	Yes	
void setObject (int, Object, int, int)	1.0	Yes	
void setObject (String, Object)	3.0	No	Throws "unsupported method" exception.
void setObject (String, Object, int)	3.0	No	Throws "unsupported method" exception.
void setObject (String, Object, int, int)	3.0	No	Throws "unsupported method" exception.
void setQueryTimeout (int)	1.0	Yes	Not supported on DB2 z/OS servers. Support for the JDBC and ODBC Socket Servers is dependent on the driver it loads.
void setRef (int, Ref)	2.0 Core	No	Throws "unsupported method" exception.
void setShort (int, short)	1.0	Yes	

CallableStatement			
Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void setShort (String, short)	3.0	No	Throws “unsupported method” exception.
void setString (int, String)	1.0	Yes	
void setString (String, String)	3.0	No	Throws “unsupported method” exception.
void setTime (int, Time)	1.0	Yes	SQLServer throws "optional feature not implemented" exception.
void setTime (int, Time, Calendar)	2.0 Core	Yes	SQLServer throws "optional feature not implemented" exception.
void setTime (String, Time)	3.0	No	Throws “unsupported method” exception.
void setTime (String, Time, Calendar)	3.0	No	Throws “unsupported method” exception.
void setTimestamp (int, Timestamp)	1.0	Yes	
void setTimestamp (int, Timestamp, Calendar)	2.0 Core	Yes	
void setTimestamp (String, Timestamp)	3.0	No	Throws “unsupported method” exception.
void setTimestamp (String, Timestamp, Calendar)	3.0	No	Throws “unsupported method” exception.
void setUnicodeStream (int, InputStream, int)	1.0	Yes	Supported by SequeLink. This method has been deprecated in the JDBC 2.0 specification.
void setURL (int, URL)	3.0	No	Throws “unsupported method” exception.
void setURL (String, URL)	3.0	No	Throws “unsupported method” exception.

Clob Object

Clob Object Methods	Version Introduced	Supported	Comments
InputStream getAsciiStream ()	2.0 Core	Yes	
Reader getCharacterStream ()	2.0 Core	Yes	
String getSubString (long, int)	2.0 Core	Yes	
long length ()	2.0 Core	Yes	
long position (Clob, long)	2.0 Core	Yes	
long position (String, long)	2.0 Core	Yes	
OutputStream setAsciiStream (long)	3.0 Core	Yes	
Writer setCharacterStream (long)	3.0 Core	Yes	
int setString (long, String)	3.0 Core	Yes	
int setString (long, String, int, int)	3.0 Core	Yes	
void truncate (long)	3.0 Core	Yes	
NOTE: Clob support is emulated using LONGVARCHAR datatypes on SQL Server, Sybase, JDBC Socket, and ODBC Socket. Using emulated Clobs incurs a performance and scalability penalty.			

Connection Object

Connection Object Methods	Version Introduced	Supported	Comments
void clearWarnings ()	1.0	Yes	

Connection Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void close ()	1.0	Yes	When a connection is closed while there is an active transaction, that transaction is rolled-back.
void commit ()	1.0	Yes	
SQLWarning getWarnings ()	1.0	Yes	
Statement createStatement ()	1.0	Yes	
Statement createStatement (int, int)	2.0 Core	Yes	The supported values for resultSetConcurrency depend on the data store you are connected to.
Statement createStatement (int, int, int)	3.0	No	SequeLink only supports fixed Holdability values per RDBMS
boolean getAutoCommit ()	1.0	Yes	
String getCatalog ()	1.0	Yes	Support is database-specific.
int getHoldability ()	3.0	Yes	SequeLink only supports fixed Holdability values per RDBMS
DatabaseMetaData getMetaData ()	1.0	Yes	
int getTransactionIsolation ()	1.0	Yes	
Map getTypeMap ()	2.0 Core	Yes	Always returns empty java.util.HashMap.
SQLWarning getWarnings ()	1.0	Yes	
boolean isClosed ()	1.0	Yes	
boolean isReadOnly ()	1.0	Yes	
String nativeSQL (String)	1.0	Yes	Always returns same String as passed in.
CallableStatement prepareCall (String)	1.0	Yes	

Connection Object <i>(cont.)</i> Methods	Version Introduced	Supported	Comments
CallableStatement prepareCall (String, int, int)	2.0 Core	Yes	The supported values for resultSetConcurrency depend on the data store you are connected to.
CallableStatement prepareCall (String, int, int, int)	3.0	No	SequeLink only supports fixed Holdability values per RDBMS
PreparedStatement prepareStatement (String)	1.0	Yes	
PreparedStatement prepareStatement (String, int)	3.0	No	Throws “unsupported method” exception.
PreparedStatement prepareStatement (String, int[])	3.0	No	Throws “unsupported method” exception.
PreparedStatement prepareStatement (String, int, int)	2.0 Core	Yes	The supported values for resultSetType and resultSetConcurrency depend on the data store you are connected to.
PreparedStatement prepareStatement (String, int, int, int)	3.0	No	SequeLink only supports fixed Holdability values per RDBMS
PreparedStatement prepareStatement (String, String [])	3.0	No	Throws “unsupported method” exception.
void releaseSavepoint (Savepoint)	3.0	Yes	
void rollback ()	1.0	Yes	
void rollback (Savepoint)	3.0	Yes	
void setAutoCommit (boolean)	1.0	Yes	

Connection Object (cont.) Methods	Version Introduced	Supported	Comments
void setCatalog (String)	1.0	Yes	Switching of catalogs is supported if the SequeLink Server supports it.
void setHoldability (int)	3.0	No	SequeLink only supports fixed Holdability values per RDBMS.
void setReadOnly (boolean)	1.0	Yes	
void setSavepoint ()	3.0	Yes	Not supported with Informix servers.
void setSavepoint (String)	3.0	Yes	Not supported with Informix servers.
void setTransactionIsolation (int)	1.0	Yes	
void setTypeMap (Map)	2.0 Core	Yes	Ignored.

ConnectionPoolDataSource Object

ConnectionPoolDataSource Object Methods	Version Introduced	Supported	Comments
PrintWriter getLogWriter ()	2.0 Optional	Yes	
int getLoginTimeout ()	2.0 Optional	Yes	
PooledConnection getPooledConnection ()	2.0 Optional	Yes	
PooledConnection getPooledConnection (String, String)	2.0 Optional	Yes	

ConnectionPoolData			
Source Object <i>(cont.)</i>	Version		
Methods	Introduced	Supported	Comments
void setLogWriter (PrintWriter)	2.0 Optional	Yes	
void setLoginTimeout (int)	2.0 Optional	Yes	

DatabaseMetaData Object

DatabaseMetaData Object	Version		
Methods	Introduced	Supported	Comments
boolean allProceduresAreCallable ()	1.0	Yes	
boolean allTablesAreSelectable ()	1.0	Yes	
boolean dataDefinitionCausesTransaction Commit ()	1.0	Yes	
boolean dataDefinitionIgnoredInTransactions ()	1.0	Yes	
boolean deletesAreDetected (int)	2.0 Core	Yes	
boolean doesMaxRowSizeIncludeBlobs ()	1.0	Yes	Not supported with DB2, SQL Server, and Sybase servers.
ResultSet getAttributes (String, String)	3.0	No	Always returns empty result set.
ResultSet getBestRowIdentifier (String, String, String, int, boolean)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.

DatabaseMetaData Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
ResultSet getCatalogs ()	1.0	Yes	This method is supported if the SequeLink Server supports it.
String getCatalogSeparator ()	1.0	Yes	
String getCatalogTerm ()	1.0	Yes	
ResultSet getColumnPrivileges (String, String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
ResultSet getColumns (String, String, String, String)	1.0	Yes	
ResultSet getColumns (String, String, String, String)	3.0 Extension	Yes	The extended resultset columns will all contain null as values.
Connection getConnection ()	2.0 Core	Yes	
ResultSet getCrossReference (String, String, String, String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
int getDatabaseMajorVersion ()	3.0	Yes	
int getDatabaseMinorVersion ()	3.0	Yes	
String getDatabaseProductName ()	1.0	Yes	
String getDatabaseProductVersion ()	1.0	Yes	
int getDefaultTransactionIsolation ()	1.0	Yes	
int getDriverMajorVersion ()	1.0	Yes	
int getDriverMinorVersion ()	1.0	Yes	

DatabaseMetaData Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
String getDriverName ()	1.0	Yes	
String getDriverVersion ()	1.0	Yes	
ResultSet getExportedKeys (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
String getExtraNameCharacters ()	1.0	Yes	
String getIdentifierQuoteString ()	1.0	Yes	
ResultSet getImportedKeys (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
ResultSet getIndexInfo (String, String, String, boolean, boolean)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
int getJDBCMajorVersion ()	3.0	Yes	
int getJDBCMinorVersion ()	3.0	Yes	
int getMaxBinaryLiteralLength ()	1.0	Yes	
int getMaxCatalogNameLength ()	1.0	Yes	
int getMaxCharLiteralLength ()	1.0	Yes	
int getMaxColumnNameLength ()	1.0	Yes	
int getMaxColumnsInGroupBy ()	1.0	Yes	
int getMaxColumnsInIndex ()	1.0	Yes	
int getMaxColumnsInOrderBy ()	1.0	Yes	
int getMaxColumnsInSelect ()	1.0	Yes	

DatabaseMetaData Object (cont.) Methods	Version Introduced	Supported	Comments
int getMaxColumnsInTable ()	1.0	Yes	
int getMaxConnections ()	1.0	Yes	
int getMaxCursorNameLength ()	1.0	Yes	
int getMaxIndexLength ()	1.0	Yes	
int getMaxProcedureNameLength ()	1.0	Yes	
int getMaxRowSize ()	1.0	Yes	
int getMaxSchemaNameLength ()	1.0	Yes	
int getMaxStatementLength ()	1.0	Yes	
int getMaxStatements ()	1.0	Yes	
int getMaxTableNameLength ()	1.0	Yes	
int getMaxTablesInSelect ()	1.0	Yes	
int getMaxUserNameLength ()	1.0	Yes	
String getNumericFunctions ()	1.0	Yes	
ResultSet getPrimaryKeys (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
ResultSet getProcedureColumns (String, String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
String getProcedureTerm ()	1.0	Yes	
ResultSet getProcedures (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store has support for it.

DatabaseMetaData Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
int getResultSetHoldability ()	3.0	Yes	
ResultSet getSchemas ()	1.0	Yes	The JDBC driver supports this method if the data store has support for it.
ResultSet getSchemas ()	3.0 Extension	Yes	The JDBC driver supports this method if the data store has support for it.
String getSchemaTerm ()	1.0	Yes	
String getSearchStringEscape ()	1.0	Yes	
String getStringFunctions ()	1.0	Yes	
String getSQLKeywords ()	1.0	Yes	
int getSQLStateType ()	3.0	Yes	
String getSystemFunctions ()	1.0	Yes	
ResultSet getSuperTables (String, String, String)	3.0	No	Always returns empty ResultSet.
ResultSet getSuperTypes (String, String, String)	3.0	No	Always returns empty ResultSet.
ResultSet getTablePrivileges (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store supports it.
ResultSet getTableTypes ()	1.0	Yes	
ResultSet getTables (String, String, String, String [])	1.0	Yes	

DatabaseMetaData Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
ResultSet getTables (String, String, String, String [])	3.0 Extension	Yes	The extended resultset columns will all contain null as values.
String getTimeDateFunctions ()	1.0	Yes	
ResultSet getTypeInfo ()	1.0	Yes	
ResultSet getUDTs (String, String, String, int [])	2.0 Core	No	SequeLink does not support the Advanced Data Type functionality and always returns an empty result set.
ResultSet getUDTs (String, String, String, int [])	3.0 Extension	No	SequeLink does not support the Advanced Data Type functionality and always returns an empty result set.
String getURL ()	1.0	Yes	
String getUsername ()	1.0	Yes	
ResultSet getVersionColumns (String, String, String)	1.0	Yes	The JDBC driver supports this method if the data store supports it.
boolean insertsAreDetected (int)	2.0 Core	Yes	
boolean isCatalogAtStart ()	1.0	Yes	
boolean isReadOnly ()	1.0	Yes	
boolean locatorsUpdateCopy ()	3.0	Yes	
boolean nullPlusNonNullIsNull ()	1.0	Yes	

DatabaseMetaData Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
boolean nullsAreSortedAtEnd ()	1.0	Yes	
boolean nullsAreSortedAtStart ()	1.0	Yes	
boolean nullsAreSortedHigh ()	1.0	Yes	
boolean nullsAreSortedLow ()	1.0	Yes	
boolean othersDeletesAreVisible (int)	2.0 Core	Yes	
boolean othersInsertsAreVisible (int)	2.0 Core	Yes	
boolean othersUpdatesAreVisible (int)	2.0 Core	Yes	
boolean ownDeletesAreVisible (int)	2.0 Core	Yes	
boolean ownInsertsAreVisible (int)	2.0 Core	Yes	
boolean ownUpdatesAreVisible (int)	2.0 Core	Yes	
boolean storesLowerCaseIdentifiers ()	1.0	Yes	
boolean storesLowerCaseQuoted Identifiers ()	1.0	Yes	
boolean storesMixedCaseIdentifiers ()	1.0	Yes	
boolean storesMixedCaseQuoted Identifiers ()	1.0	Yes	
boolean storesUpperCaseIdentifiers ()	1.0	Yes	
boolean supportsResultSetHoldability (int)	3.0	Yes	
boolean supportsAlterTableWith AddColumn ()	1.0	Yes	
boolean supportsAlterTableWith DropColumn ()	1.0	Yes	
boolean supportsANSI92EntryLevelSQL ()	1.0	Yes	
boolean supportsANSI92FullSQL ()	1.0	Yes	
boolean supportsANSI92Intermediate SQL ()	1.0	Yes	

DatabaseMetaData Object (cont.) Methods	Version Introduced	Supported	Comments
boolean supportsBatchUpdates ()	2.0 Core	Yes	
boolean supportsCatalogsInData Manipulation ()	1.0	Yes	
boolean supportsCatalogsInIndex ()	1.0	Yes	
boolean supportsCatalogsInPrivilege Definitions ()	1.0	Yes	
boolean supportsCatalogsInProcedure Calls ()	1.0	Yes	
boolean supportsCatalogsInTable Definitions ()	1.0	Yes	
boolean supportsColumnAliasing ()	1.0	Yes	
boolean supportsConvert ()	1.0	Yes	
boolean supportsConvert (int, int)	1.0	Yes	
boolean supportsCoreSQLGrammar ()	1.0	Yes	
boolean supportsCorrelatedSubqueries ()	1.0	Yes	
boolean supportsDataDefinitionAndData ManipulationTransactions ()	1.0	Yes	
boolean supportsDataManipulation TransactionsOnly ()	1.0	Yes	
boolean supportsDifferentTableCorrelation Names ()	1.0	Yes	
boolean supportsExpressionsIn OrderBy ()	1.0	Yes	
boolean supportsExtendedSQLGrammar ()	1.0	Yes	
boolean supportsFullOuterJoins ()	1.0	Yes	
boolean supportsGetGeneratedKeys ()	3.0	Yes	
boolean supportsGroupBy ()	1.0	Yes	

DatabaseMetaData Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
boolean supportsGroupByBeyondSelect ()	1.0	Yes	
boolean supportsGroupByUnrelated ()	1.0	Yes	
boolean supportsIntegrityEnhancement Facility ()	1.0	Yes	
boolean supportsLikeEscapeClause ()	1.0	Yes	
boolean supportsLimitedOuterJoins ()	1.0	Yes	
boolean supportsMinimumSQLGrammar ()	1.0	Yes	
boolean supportsMixedCaseIdentifiers ()	1.0	Yes	
boolean supportsMixedCaseQuoted Identifiers ()	1.0	Yes	
boolean supportsMultipleOpenResultSets ()	3.0	Yes	
boolean supportsMultipleResultSets ()	1.0	Yes	Not supported on Informix. NOTE: Use of this method with large result sets incurs a performance and scalability penalty.
boolean supportsMultipleTransactions ()	1.0	Yes	
boolean supportsNamedParameters ()	3.0	Yes	
boolean supportsNonNullableColumns ()	1.0	Yes	
boolean supportsOpenCursorsAcross Commit ()	1.0	Yes	
boolean supportsOpenCursorsAcross Rollback ()	1.0	Yes	
boolean supportsOpenStatementsAcross Commit ()	1.0	Yes	

DatabaseMetaData Object (cont.) Methods	Version Introduced	Supported	Comments
boolean supportsOpenStatementsAcross Rollback ()	1.0	Yes	
boolean supportsOrderByUnrelated ()	1.0	Yes	
boolean supportsOuterJoins ()	1.0	Yes	
boolean supportsPositionedDelete ()	1.0	Yes	The driver returns hard coded false.
boolean supportsPositionedUpdate ()	1.0	Yes	The driver returns hard coded false.
boolean supportsResultSetConcurrency (int, int)	2.0 Core	Yes	
boolean supportsResultSetType (int)	2.0 Core	Yes	
boolean supportsSavePoints ()	3.0	Yes	
boolean supportsSchemasInData Manipulation ()	1.0	Yes	
boolean supportsSchemasInIndex Definitions ()	1.0	Yes	
boolean supportsSchemasIn PrivilegeDefinitions ()	1.0	Yes	
boolean supportsSchemasInProcedure Calls ()	1.0	Yes	
boolean supportsSchemasInTable Definitions ()	1.0	Yes	

DatabaseMetaData Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
boolean supportsSelectForUpdate ()	1.0	Yes	SequeLink supports SELECT FOR UPDATE statements against certain DBMSs. SequeLink returns false, as it does not support UPDATE WHERE CURRENT OF statements.
boolean supportsStoredProcedures ()	1.0	Yes	
boolean supportsSubqueriesIn Comparisons ()	1.0	Yes	
boolean supportsSubqueriesInExists ()	1.0	Yes	
boolean supportsSubqueriesInIns ()	1.0	Yes	
boolean supportsSubqueriesIn Quantifieds ()	1.0	Yes	
boolean supportsTableCorrelationNames ()	1.0	Yes	
boolean supportsTransactionIsolationLevel (int)	1.0	Yes	
boolean supportsTransactions ()	1.0	Yes	
boolean supportsUnion ()	1.0	Yes	
boolean supportsUnionAll ()	1.0	Yes	
boolean updatesAreDetected (int)	2.0 Core	Yes	
boolean usesLocalFilePerTable ()	1.0	Yes	
boolean usesLocalFiles ()	1.0	Yes	

DataSource Object

DataSource Object Methods	Version Introduced	Supported	Comments
Connection getConnection ()	2.0 Optional	Yes	
Connection getConnection (String, String)	2.0 Optional	Yes	
PrintWriter getLogWriter ()	2.0 Optional	Yes	
int getLoginTimeout ()	2.0 Optional	Yes	
void setLogWriter (PrintWriter)	2.0 Optional	Yes	Enables DataDirect Spy, which traces JDBC information into the specified PrintWriter.
void setLoginTimeout (int)	2.0 Optional	Yes	

Driver Object

Driver Object Methods	Version Introduced	Supported	Comments
boolean acceptsURL (String)	1.0	Yes	
Connection connect (String, Properties)	1.0	Yes	
int getMajorVersion ()	1.0	Yes	
int getMinorVersion ()	1.0	Yes	
DriverPropertyInfo [] getPropertyInfo (String, Properties)	1.0	Yes	
boolean jdbcCompliant ()	1.0	Yes	

ParameterMetaData Object

ParameterMetaData Object Methods	Version Introduced	Supported	Comments
String getParameterClassName (int)	3.0	Yes	
int getParameterCount ()	3.0	Yes	
int getParameterMode (int)	3.0	Yes	
int getParameterType (int)	3.0	Yes	
String getParameterTypeName (int)	3.0	Yes	
int getPrecision (int)	3.0	Yes	
int getScale (int)	3.0	Yes	
int isNullable (int)	3.0	Yes	
boolean isSigned (int)	3.0	Yes	

PooledConnection Object

PooledConnection Object Methods	Version Introduced	Supported	Comments
void addConnectionEventListener (ConnectionEventListener)	2.0 Optional	Yes	
void close()	2.0 Optional	Yes	

PooledConnection Object			
(cont.)	Version		
Methods	Introduced	Supported	Comments
Connection getConnection()	2.0 Optional	Yes	A pooled connection object can have only one Connection object open (the one most recently created). The purpose of allowing the server (PoolManager implementation) to invoke the method getConnection a second time is to give that application server a way to take a connection away from an application and give it to another user (a rare occurrence). The drivers do not support this "reclaiming" of connections and will throw a SQLException "Reclaim of open connection is not supported."
void removeConnectionEventListener (ConnectionEventListener)	2.0 Optional	Yes	

PreparedStatement Object

PreparedStatement Object			
	Version		
Methods	Introduced	Supported	Comments
void addBatch ()	2.0 Core	Yes	
void addBatch (String)	2.0 Core	No	Throws "invalid method call" exception.

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
void cancel ()	1.0	Yes	Cancel can be used to cancel a function running synchronously on a statement, using a different thread. Whether Cancel will actually cancel the running function depends on the data store.
void clearBatch ()	2.0 Core	Yes	
void clearParameters ()	1.0	Yes	
void clearWarnings ()	1.0	Yes	
void close ()	1.0	Yes	
boolean execute ()	1.0	Yes	
boolean execute (String)	1.0	No	Throws "invalid method call" exception.
boolean execute (String, int)	3.0	No	Throws "invalid method call" exception.
boolean execute (String, int [])	3.0	No	Throws "invalid method call" exception.
boolean execute (String, String [])	3.0	No	Throws "invalid method call" exception.
int [] executeBatch ()	2.0 Core	Yes	
ResultSet executeQuery ()	1.0	Yes	
ResultSet executeQuery (String)	1.0	No	Throws "invalid method call" exception.
int executeUpdate ()	1.0	Yes	
int executeUpdate (String)	1.0	No	Throws "invalid method call" exception.
int executeUpdate (String, int)	3.0	No	Throws "unsupported method" exception.

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
int executeUpdate (String, int [])	3.0	No	Throws “unsupported method” exception.
int executeUpdate (String, String [])	3.0	No	Throws “unsupported method” exception.
Connection getConnection ()	2.0 Core	Yes	
int getFetchDirection ()	2.0 Core	Yes	
int getFetchSize ()	2.0 Core	Yes	
ParameterMetaData getParameterMetaData()	3.0	Yes	Not supported on Oracle or Informix Servers.
ResultSet getGeneratedKeys ()	3.0	No	Throws “unsupported method” exception.
int getMaxFieldSize ()	1.0	Yes	
int getMaxRows ()	1.0	Yes	
ResultSetMetaData getMetaData ()	2.0 Core	Yes	
boolean getMoreResults ()	1.0	Yes	
boolean getMoreResults (int)	3.0	Yes	
int getQueryTimeout ()	1.0	Yes	
ResultSet getResultSet ()	1.0	Yes	
int getResultSetConcurrency ()	2.0 Core	Yes	
int getResultSetHoldability ()	3.0	No	Throws “unsupported method” exception.
int getResultSetType ()	2.0 Core	Yes	
int getUpdateCount ()	1.0	Yes	
SQLWarning getWarnings ()	1.0	Yes	

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
void setArray (int, Array)	2.0 Core	No	Throws "unsupported method" exception.
void setAsciiStream (int, InputStream, int)	1.0	Yes	
void setBigDecimal (int, BigDecimal)	1.0	Yes	
void setBinaryStream (int, InputStream, int)	1.0	Yes	
void setBlob (int, Blob)	2.0 Core	Yes	
void setBoolean (int, boolean)	1.0	Yes	
void setByte (int, byte)	1.0	Yes	
void setBytes (int, byte [])	1.0	Yes	
void setCharacterStream (int, Reader, int)	2.0 Core	Yes	
void setClob (int, Clob)	2.0 Core	Yes	
void setCursorName (String)	1.0	No	
void setDate (int, Date)	1.0	Yes	SQL Server throws "optional feature not implemented" exception.
void setDate (int, Date, Calendar)	2.0 Core	Yes	SQL Server throws "optional feature not implemented" exception.
void setDouble (int, double)	1.0	Yes	
void setEscapeProcessing (boolean)	1.0	Yes	Ignored.
void setFetchDirection (int)	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
void setFetchSize (int)	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.
void setFloat (int, float)	1.0	Yes	
void setInt (int, int)	1.0	Yes	
void setLong (int, long)	1.0	Yes	
void setMaxFieldSize (int)	1.0	Yes	
void setMaxRows (int)	1.0	Yes	
void setNull (int, int)	1.0	Yes	
void setNull (int, int, String)	2.0 Core	Yes	SequeLink does not support the Advanced Data Type functionality. This method is always mapped to setNull (i, sqlType).
void setObject (int, Object)	1.0	Yes	
void setObject (int, Object, int)	1.0	Yes	
void setObject (int, Object, int, int)	1.0	Yes	
void setQueryTimeout (int)	1.0	Yes	Support for the JDBC and ODBC Socket Servers is dependent on the driver it loads.
void setRef (int, Ref)	2.0 Core	No	Throws “unsupported method” exception.
void setShort (int, short)	1.0	Yes	
void setString (int, String)	1.0	Yes	

PreparedStatement Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
void setTime (int, Time)	1.0	Yes	SQL Server throws "optional feature not implemented" exception.
void setTime (int, Time, Calendar)	2.0 Core	Yes	SQL Server throws "optional feature not implemented" exception.
void setTimestamp (int, Timestamp)	1.0	Yes	
void setTimestamp (int, Timestamp, Calendar)	2.0 Core	Yes	
void setUnicodeStream (int, InputStream, int)	1.0	Yes	Throws "unsupported method" exception. This method was deprecated in the JDBC 2.0 specification.
void setURL (int, URL)	3.0	No	Throws "unsupported method" exception.
boolean wasNull ()	1.0	Yes	

Ref Object

Ref Object Methods	Version Introduced	Supported	Comments
(all)	2.0 Core	No	Ref objects are neither exposed, nor taken as input.

Referenceable Object

Referenceable Object	JDBC Version Introduced	Supported	Comments
Methods			
Reference getReference()	javax.naming	Yes	Implemented by SequeLinkDataSource.

ResultSet Object

ResultSet Object	Version Introduced	Supported	Comments
Methods			
boolean absolute (int)	2.0 Core	Yes	
void afterLast ()	2.0 Core	Yes	
void beforeFirst ()	2.0 Core	Yes	
void cancelRowUpdates ()	2.0 Core	Yes	
void clearWarnings ()	1.0	Yes	
void close ()	1.0	Yes	
void deleteRow ()	2.0 Core	Yes	
int findColumn (String)	1.0	Yes	
boolean first ()	2.0 Core	Yes	
Array getArray (int)	2.0 Core	No	Throws “unsupported method” exception.
Array getArray (String)	2.0 Core	No	Throws “unsupported method” exception.

ResultSet Object <i>(cont.)</i>	Version		
Methods	Introduced	Supported	Comments
InputStream getAsciiStream (int)	1.0	Yes	For the implementation of getAsciiStream (), the driver assumes that the underlying data in the database only contains 7-bit ASCII characters (represented in an ASCII or EBCDIC code-page). If the assumption is true, the driver returns bytes corresponding to ASCII characters, otherwise the results are unpredictable. In the case that getAsciiStream () is called on a binary column, the driver converts the binary data to the hexadecimal representation and returns the ASCII representation. For example, the binary value 20 is represented in hexadecimal as 14 or "1" "4". In ASCII, 1 is represented by "49" and 4 is represented by "52". This implies that the ASCII stream "4952" is returned.

ResultSet Object <i>(cont.)</i>	Version		
Methods	Introduced	Supported	Comments
InputStream getAsciiStream (String)	1.0	Yes	For the implementation of getAsciiStream (), the drivers assume that the underlying data in the database only contains 7-bit ASCII characters (represented in an ASCII or EBCDIC code-page). If the assumption is true, the drivers return bytes corresponding to ASCII characters, otherwise the results are unpredictable. In the case that getAsciiStream () is called on a binary column, the driver converts the binary data to the hexadecimal representation and returns the ASCII representation. For example, the binary value 20 is represented in hexadecimal as 14 or "1" "4". In ASCII, 1 is represented by "49" and 4 is represented by "52". This implies that the ASCII stream "4952" is returned.
BigDecimal getBigDecimal (int)	2.0 Core	Yes	
BigDecimal getBigDecimal (String)	2.0 Core	Yes	
BigDecimal getBigDecimal (int, int)	1.0	Yes	
BigDecimal getBigDecimal (String, int)	1.0	Yes	
InputStream getBinaryStream (int)	1.0	Yes	

ResultSet Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
InputStream getBinaryStream (String)	1.0	Yes	
Blob getBlob (int)	2.0 Core	Yes	
Blob getBlob (String)	2.0 Core	Yes	
boolean getBoolean (int)	1.0	Yes	
boolean getBoolean (String)	1.0	Yes	
byte getByte (int)	1.0	Yes	
byte getByte (String)	1.0	Yes	
byte [] getBytes (int)	1.0	Yes	
byte [] getBytes (String)	1.0	Yes	
Reader getCharacterStream (int)	2.0 Core	Yes	
Reader getCharacterStream (String)	2.0 Core	Yes	
Clob getClob (int)	2.0 Core	Yes	
Clob getClob (String)	2.0 Core	Yes	
int getConcurrency ()	2.0 Core	Yes	
String getCursorName ()	1.0	No	Throws “unsupported method” exception.
Date getDate (int)	1.0	Yes	
Date getDate (String)	1.0	Yes	
Date getDate (int, Calendar)	2.0 Core	Yes	
Date getDate (String, Calendar)	2.0 Core	Yes	
double getDouble (int)	1.0	Yes	
double getDouble (String)	1.0	Yes	

ResultSet Object <i>(cont.)</i> Methods	Version Introduced	Supported	Comments
int getFetchDirection ()	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.
int getFetchSize ()	2.0 Core	Yes	SequeLink supports this method, but will ignore this hint provided by the application.
float getFloat (int)	1.0	Yes	
float getFloat (String)	1.0	Yes	
int getInt (int)	1.0	Yes	
int getInt (String)	1.0	Yes	
long getLong (int)	1.0	Yes	
long getLong (String)	1.0	Yes	
ResultSetMetaData getMetaData ()	1.0	Yes	
Object getObject (int)	1.0	Yes	
Object getObject (String)	1.0	Yes	
Object getObject (int, Map)	2.0 Core	Yes	Map ignored.
Object getObject (String, Map)	2.0 Core	Yes	Map ignored.
Ref getRef (int)	2.0 Core	No	Throws “unsupported method” exception.
Ref getRef (String)	2.0 Core	No	Throws “unsupported method” exception.
int getRow ()	2.0 Core	Yes	
short getShort (int)	1.0	Yes	
short getShort (String)	1.0	Yes	

ResultSet Object <i>(cont.)</i>	Version Introduced	Supported	Comments
Methods			
Statement getStatement ()	2.0 Core	Yes	
String getString (int)	1.0	Yes	
String getString (String)	1.0	Yes	
Time getTime (int)	1.0	Yes	
Time getTime (String)	1.0	Yes	
Time getTime (int, Calendar)	2.0 Core	Yes	
Time getTime (String, Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (int)	1.0	Yes	
Timestamp getTimestamp (String)	1.0	Yes	
Timestamp getTimestamp (int, Calendar)	2.0 Core	Yes	
Timestamp getTimestamp (String, Calendar)	2.0 Core	Yes	
int getType ()	2.0 Core	Yes	
InputStream getUnicodeStream (int)	1.0	Yes	
InputStream getUnicodeStream (String)	1.0	Yes	This method was deprecated in the JDBC 2.0 specification.
URL getURL (int)	3.0	No	Throws “unsupported method” exception.
URL getURL (String)	3.0	No	Throws “unsupported method” exception.
SQLWarning getWarnings ()	1.0	Yes	
void insertRow ()	2.0 Core	Yes	
boolean isAfterLast ()	2.0 Core	Yes	

ResultSet Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
boolean <code>isBeforeFirst ()</code>	2.0 Core	Yes	
boolean <code>isFirst ()</code>	2.0 Core	Yes	
boolean <code>isLast ()</code>	2.0 Core	Yes	
boolean <code>last ()</code>	2.0 Core	Yes	
void <code>moveToCurrentRow ()</code>	2.0 Core	Yes	
void <code>moveToInsertRow ()</code>	2.0 Core	Yes	
boolean <code>next ()</code>	1.0	Yes	
boolean <code>previous ()</code>	2.0 Core	Yes	
void <code>refreshRow ()</code>	2.0 Core	Yes	
boolean <code>relative (int)</code>	2.0 Core	Yes	
boolean <code>rowDeleted ()</code>	2.0 Core	Yes	
boolean <code>rowInserted ()</code>	2.0 Core	Yes	
boolean <code>rowUpdated ()</code>	2.0 Core	Yes	
void <code>setFetchDirection (int)</code>	2.0 Core	Yes	
void <code>setFetchSize (int)</code>	2.0 Core	Yes	
void <code>updateArray (int, Array)</code>	3.0	No	Throws “unsupported method” exception.
void <code>updateArray (String, Array)</code>	3.0	No	Throws “unsupported method” exception.
void <code>updateAsciiStream (int, InputStream, int)</code>	2.0 Core	Yes	
void <code>updateAsciiStream (String, InputStream, int)</code>	2.0 Core	Yes	
void <code>updateBigDecimal (int, BigDecimal)</code>	2.0 Core	Yes	
void <code>updateBigDecimal (String, BigDecimal)</code>	2.0 Core	Yes	

ResultSet Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void updateBinaryStream (int, InputStream, int)	2.0 Core	Yes	
void updateBinaryStream (String, InputStream, int)	2.0 Core	Yes	
void updateBlob (int, Blob)	3.0	Yes	
void updateBlob (String, Blob)	3.0	Yes	
void updateBoolean (int, boolean)	2.0 Core	Yes	
void updateBoolean (String, boolean)	2.0 Core	Yes	
void updateByte (int, byte)	2.0 Core	Yes	
void updateByte (String, byte)	2.0 Core	Yes	
void updateBytes (int, byte [])	2.0 Core	Yes	
void updateBytes (String, byte [])	2.0 Core	Yes	
void updateCharacterStream (int, Reader, int)	2.0 Core	Yes	
void updateCharacterStream (String, Reader, int)	2.0 Core	Yes	
void updateClob (int, Clob)	3.0	No	
void updateClob (String, Clob)	3.0	No	
void updateDate (int, Date)	2.0 Core	Yes	
void updateDate (String, Date)	2.0 Core	Yes	
void updateDouble (int, double)	2.0 Core	Yes	
void updateDouble (String, double)	2.0 Core	Yes	

ResultSet Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void updateFloat (int, float)	2.0 Core	Yes	
void updateFloat (String, float)	2.0 Core	Yes	
void updateInt (int, int)	2.0 Core	Yes	
void updateInt (String, int)	2.0 Core	Yes	
void updateLong (int, long)	2.0 Core	Yes	
void updateLong (String, long)	2.0 Core	Yes	
void updateNull (int)	2.0 Core	Yes	
void updateNull (String)	2.0 Core	Yes	
void updateObject (int, Object)	2.0 Core	Yes	
void updateObject (String, Object)	2.0 Core	Yes	
void updateObject (int, Object, int)	2.0 Core	Yes	
void updateObject (String, Object, int)	2.0 Core	Yes	
void updateRef (int, Ref)	3.0	No	Throws “unsupported method” exception.
void updateRef (String, Ref)	3.0	No	Throws “unsupported method” exception.
void updateRow ()	2.0 Core	Yes	
void updateShort (int, short)	2.0 Core	Yes	
void updateShort (String, short)	2.0 Core	Yes	
void updateString (int, String)	2.0 Core	Yes	

ResultSet Object <i>(cont.)</i>	Version	Supported	Comments
Methods	Introduced		
void updateString (String, String)	2.0 Core	Yes	
void updateTime (int, Time)	2.0 Core	Yes	
void updateTime (String, Time)	2.0 Core	Yes	
void updateTimestamp (int, Timestamp)	2.0 Core	Yes	
void updateTimestamp (String, Timestamp)	2.0 Core	Yes	
boolean wasNull ()	1.0	Yes	

ResultSetMetaData Object

ResultSetMetaData Object	Version	Supported	Comments
Methods	Introduced		
String getCatalogName (int)	1.0	Yes	The behavior of this method is data store specific.
String getColumnClassName (int)	2.0 Core	Yes	
int getColumnCount ()	1.0	Yes	
int getColumnDisplaySize (int)	1.0	Yes	
String getColumnLabel (int)	1.0	Yes	
String getColumnName (int)	1.0	Yes	
int getColumnType (int)	1.0	Yes	
String getColumnName (int)	1.0	Yes	
int getPrecision (int)	1.0	Yes	

ResultSetMetaData			
Object <i>(cont.)</i>	Version		
Methods	Introduced	Supported	Comments
int getScale (int)	1.0	Yes	
String getSchemaName (int)	1.0	Yes	The behavior of this method is data store specific.
String getTableName (int)	1.0	Yes	The behavior of this method is data store specific.
boolean isAutoIncrement (int)	1.0	Yes	
boolean isCaseSensitive (int)	1.0	Yes	
boolean isCurrency (int)	1.0	Yes	
boolean isDefinitelyWritable (int)	1.0	Yes	
int isNullable (int)	1.0	Yes	
boolean isReadOnly (int)	1.0	Yes	
boolean isSearchable (int)	1.0	Yes	
boolean isSigned (int)	1.0	Yes	
boolean isWritable (int)	1.0	Yes	

RowSet Object

RowSet Object	Version		
Methods	Introduced	Supported	Comments
(all)	2.0 Optional	No	

SavePoint Object

SavePoint Object Methods	Version Introduced	Supported	Comments
int getSavepointId ()	3.0	Yes	Savepoint not supported on Informix, JDBC Socket, and ODBC Socket Server servers.
String getSavepointName ()	3.0	Yes	Savepoint not supported on Informix, JDBC Socket, and ODBC Socket Server servers.

Serializable Object

Serializable Object Methods	Version Introduced	Supported	Comments
(N/A)	java.io	Yes	Implemented by SequeLinkDataSource.

Statement Object

Statement Object Methods	Version Introduced	Supported	Comments
void addBatch (String)	2.0 Core	Yes	
void cancel ()	1.0	Yes	Cancel can be used to cancel a function running synchronously on a statement, using a different thread. Whether Cancel will actually cancel the running function depends on the data store.
void clearBatch ()	2.0 Core	Yes	

Statement Object <i>(cont.)</i> Methods	Version Introduced	Supported	Comments
void clearWarnings ()	1.0	Yes	
void close ()	1.0	Yes	
boolean execute (String)	1.0	Yes	
boolean execute (String, int)	3.0	No	Throws “unsupported method” exception.
boolean execute (String, int [])	3.0	No	Throws “unsupported method” exception.
boolean execute (String, String [])	3.0	No	Throws “unsupported method” exception.
int [] executeBatch ()	2.0 Core	Yes	
ResultSet executeQuery (String)	1.0	Yes	
int executeUpdate (String)	1.0	Yes	
int executeUpdate (String, int)	3.0	No	Throws “unsupported method” exception.
int executeUpdate (String, int [])	3.0	No	Throws “unsupported method” exception.
int executeUpdate (String, String [])	3.0	No	Throws “unsupported method” exception.
Connection getConnection ()	2.0 Core	Yes	
int getFetchDirection ()	2.0 Core	Yes	SequeLink supports this method, but ignores this hint provided by the application.
int getFetchSize ()	2.0 Core	Yes	SequeLink supports this method, but ignores this hint provided by the application.
ResultSet getGeneratedKeys ()	3.0	No	Throws “unsupported method” exception.
int getMaxFieldSize ()	1.0	Yes	

Statement Object <i>(cont.)</i> Methods	Version Introduced	Supported	Comments
int getMaxRows ()	1.0	Yes	
boolean getMoreResults ()	1.0	Yes	
boolean getMoreResults (int)	3.0	No	Throws “unsupported method” exception.
int getQueryTimeout ()	1.0	Yes	Returns 0 for DB2 UDB and Informix.
ResultSet getResultSet ()	1.0	Yes	
int getResultSetConcurrency ()	2.0 Core	Yes	
int getResultSetHoldability ()	3.0	No	Throws “unsupported method” exception.
int getResultSetType ()	2.0 Core	Yes	
int getUpdateCount ()	1.0	Yes	
SQLWarning getWarnings ()	1.0	Yes	
void setCursorName (String)	1.0	No	
void setEscapeProcessing (boolean)	1.0	Yes	
void setFetchDirection (int)	2.0 Core	Yes	
void setFetchSize (int)	2.0 Core	Yes	
void setMaxFieldSize (int)	1.0	Yes	
void setMaxRows (int)	1.0	Yes	
void setQueryTimeout (int)	1.0	Yes	Support for JDBC and ODBC Socket Servers is dependent on the driver it loads.

Struct Object

Struct Object Methods	Version Introduced	Supported	Comments
(all)	2.0	No	Struct objects are neither exposed, nor taken as input.

XAConnection Object

XAConnection Object Methods	Version Introduced	Supported	Comments
XAResource getXAResource()	2.0 Optional	Yes	

XADataSource Object

XADataSource Object Methods	Version Introduced	Supported	Comments
int getLoginTimeout ()	2.0 Optional	Yes	
PrintWriter getLogWriter ()	2.0 Optional	Yes	

XADataSource Object (cont.)			
Methods	Version Introduced	Supported	Comments
XAConnection getXAConnection ()	2.0 Optional	Yes	The behavior of these methods depends on whether distributed/XA transactions are supported by the SequeLink Server. If not supported, the JDBC driver will throw an "XA-Open failed with return code -3" exception.
XAConnection getXAConnection (String, String)	2.0 Optional	Yes	The behavior of these methods depends on whether distributed/XA transactions are supported by the SequeLink Server. If not supported, the JDBC driver will throw an "XA-Open failed with return code -3" exception.
void setLoginTimeout (int)	2.0 Optional	Yes	
void setLogWriter (PrintWriter)	2.0 Optional	Yes	

D JDBC Connection Pool Manager

Connection pooling means that connections are reused rather than created each time a connection is requested. Your application can use connection pooling through the DataDirect Connection Pool Manager.

Connection pooling is performed in the background and does not affect how an application is coded; however, the application must use a DataSource object (an object implementing the DataSource interface) to obtain a connection instead of using the DriverManager class. A class implementing the DataSource interface may or may not provide connection pooling. A DataSource object registers with a JNDI naming service. Once a DataSource object is registered, the application retrieves it from the JNDI naming service in the standard way.

Creating a Data Source

This section contains sample code that is provided as an example of using the DataDirect Connection Pool Manager to allow your applications to handle connection pooling.

Creating a DataDirect SequeLink® Data Source Object

The following example shows how to create a SequeLink *for* JDBC DataSource object and register it to a JNDI naming service. The DataSource class is provided by your SequeLink *for* JDBC driver and is database-independent. In the following example we use Oracle, so the DataSource class is SequeLinkDataSource. See [Chapter 8 “Developing JDBC Applications” on page 327](#) for the name of the DataSource class.

If you want the client application to use *non-pooled* connections (see [“Connecting to a Data Source” on page 547](#)), you must modify this example so that the JNDI entry is registered using the name `jdbc/SparkyOracle`.

If you want the client application to use *pooled* connections, the JNDI entry must map to the DataSource of the DataDirect Connection Pool Manager. Therefore, you must register two data sources:

- The Connection Pool Manager's Data Source using the example in [“Creating a Data Source Using the DataDirect® Connection Pool Manager” on page 544](#). This process registers the data source using the JNDI entry `jdbc/SparkyOracle`. The Connection Pool Manager will create physical connections using the JNDI Entry `jdbc/SequeLinkSparkyOracle`.
- A SequeLink DataSource, using the following example to register the DataSource using the JNDI entry `jdbc/SequeLinkSparkyOracle`.

```

//*****
//
// This code creates a SequeLink for JDBC data source and registers it to a
// JNDI naming service. This SequeLink for JDBC data source uses the
// DataSource implementation provided by the SequeLink for JDBC Driver.
//
// If you want users to use non-pooled connections, you must modify this
// example so that it registers the SequeLink Data Source using the JNDI
// entry <jdbc/SparkyOracle>.
//
// If you want users to use pooled connections, use this example as is
// to register the SequeLink Data Source using the JNDI entry
// <jdbc/SequeLinkSparkyOracle>. Also, use the example in the next section
// to register the Connection Pool Manager's Data Source using the JNDI entry
// <jdbc/SparkyOracle>
//
//*****

// From SequeLink for JDBC:
import com.ddtek.jdbcx.sequelink.SequeLinkDataSource;

import javax.sql.*;
import java.sql.*;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.Hashtable;

public class SequeLinkDataSourceRegisterJNDI
{
    public static void main(String argv[])
    {
        try {
            // Set up data source reference data for naming context:
            // -----
            // Create a class instance that implements the interface
            // ConnectionPoolDataSource
            OracleDataSource ds = new SequeLinkDataSource();

            ds.setDescription(
                "Oracle on Sparky - SequeLink Data Source");
            ds.setServerName("sparky");
        }
    }
}

```

```

        ds.setPortNumber(19996);
        ds.setUser("scott");
        ds.setPassword("test");

        // Set up environment for creating initial context
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
            "com.sun.jndi.fscontext.RefFSContextFactory");
        env.put(Context.PROVIDER_URL, "file:c:\\JDBCDataSource");
        Context ctx = new InitialContext(env);

        // Register the data source to JNDI naming service
        ctx.bind("jdbc/SequeLinkSparkyOracle", ds);

    } catch (Exception e) {
        System.out.println(e);
        return;
    }
} // Main
} // class SequeLinkDataSourceRegisterJNDI

```

Creating a Data Source Using the DataDirect® Connection Pool Manager

The following Java code example creates a data source for JDBC and registers it to a JNDI naming service. The `PooledConnectionDataSource` class is provided by the DataDirect `com.ddtek.pool` package. In the following code example, the `PooledConnectionDataSource` object references a JDBC data source object. Therefore, the example performs a lookup by setting the `DataSourceName` attribute to the JNDI name of a registered pooled data source (in this example, `jdbc/SequeLinkSparkyOracle`, which is the JDBC `DataSource` object created in section [“Creating a DataDirect SequeLink® Data Source Object”](#) on page 542).

Client applications that use this data source must perform a lookup using the registered JNDI name (jdbc/SparkyOracle in this example).

```
//*****
//
// This code creates a data source and registers it to a JNDI naming
// service. This data source uses the PooledConnectionDataSource
// implementation provided by the DataDirect com.ddtek.pool package.
//
// This data source refers to a previously registered pooled data source.
//
// This data source registers its name as <jdbc/SparkyOracle>.
// Client applications using pooling must perform a lookup for this name.
//
//*****

// From the DataDirect connection pooling package:
import com.ddtek.pool.PooledConnectionDataSource;

import javax.sql.*;
import java.sql.*;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.Hashtable;

public class PoolMgrDataSourceRegisterJNDI
{
    public static void main(String argv[])
    {
        try {
            // Set up data source reference data for naming context:
            // -----
            // Create a pooling manager's class instance that implements
            // the interface DataSource
            PooledConnectionDataSource ds = new PooledConnectionDataSource();

            ds.setDescription("Sparky Oracle - Oracle Data Source");

            // Refer to a previously registered pooled data source to access
            // a ConnectionPoolDataSource object
```

```

ds.setDataSourceName("jdbc/SequeLinkSparkyOracle");

// The pool manager will be initiated with 5 physical connections
ds.setInitialPoolSize(5);

// The pool maintenance thread will make sure that there are
// at least 5 physical connections available
ds.setMinPoolSize(5);

// The pool maintenance thread will check that there are no more
// than 10 physical connections available
ds.setMaxPoolSize(10);

// The pool maintenance thread will wake up and check the pool
// every 20 seconds
ds.setPropertyCycle(20);

// The pool maintenance thread will remove physical connections
// that are inactive for more than 300 seconds
ds.setMaxIdleTime(300);

// Set tracing off since we choose not to see output listing
// of activities on a connection
ds.setTracing(false);

// Set up environment for creating initial context
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.fscontext.RefFSContextFactory");
env.put(Context.PROVIDER_URL, "file:c:\\JDBCDataSource");
Context ctx = new InitialContext(env);

// Register the data source to JNDI naming service
// for application to use
ctx.bind("jdbc/SparkyOracle", ds);

} catch (Exception e) {
    System.out.println(e);
    return;
}

```

```

    } // Main
} // class PoolMgrDataSourceRegisterJNDI

```

Connecting to a Data Source

Because an application uses connection pooling by referencing the JNDI name of a registered `PooledConnectionDataSource` object, code changes are not required for an application to use connection pooling.

The following example shows Java code that looks up and uses the JNDI-registered data source for connections. You specify the JNDI lookup name for the data source you created (as described in [“Creating a Data Source Using the DataDirect® Connection Pool Manager”](#) on page 544).

```

//*****
//
// Test program to look up and use a JNDI-registered data source.
//
// To run the program, specify the JNDI lookup name for the
// command-line argument, for example:
//
//      java TestDataSourceApp
//
//*****
import javax.sql.*;
import java.sql.*;
import javax.naming.*;
import java.util.Hashtable;

public class TestDataSourceApp
{
    public static void main(String argv[])
    {
        String str JNDILookupName = "jdbc/SparkyOracle";

```

```

// Hard-code the JNDI entry, the application does not need to change

DataSource ds = null;
Connection con = null;
Context ctx = null;
Hashtable env = null;

long nStartTime, nStopTime, nElapsedTime;

// Set up environment for creating InitialContext object
env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.fscontext.RefFSContextFactory");
env.put(Context.PROVIDER_URL, "file:c:\\JDBCDataSource");

try {
    // Retrieve the DataSource object that bound to the logical
    // lookup JNDI name
    ctx = new InitialContext(env);
    ds = (DataSource) ctx.lookup(strJNDILookupName);
} catch (NamingException eName) {
    System.out.println("Error looking up " +
        strJNDILookupName + ": " + eName);
    System.exit(0);
}

int numOfTest = 4;
int [] nCount = {100, 100, 1000, 3000};

for (int i = 0; i < numOfTest; i++) {
    // Log the start time
    nStartTime = System.currentTimeMillis();
    for (int j = 1; j <= nCount[i]; j++) {
        // Get Database Connection
        try {
            con = ds.getConnection("scott", "tiger");
            // Do something with the connection
            // ...

            // Close Database Connection
            if (con != null) con.close();
        }
    }
}

```

```

        } catch (SQLException eCon) {
            System.out.println("Error getting a connection: " + eCon);
            System.exit(0);
        } // try getConnection
    } // for j loop

    // Log the end time
    nStopTime = System.currentTimeMillis();

    // Compute elapsed time
    nElapsedTime = nStopTime - nStartTime;
    System.out.println("Test number " + i + ": looping " +
        nCount[i] + " times");
    System.out.println("Elapsed Time: " + nElapsedTime + "\n");
} // for i loop

// All done
System.exit(0);

} // Main
} // TestDataSourceApp

```

NOTE: The JDBC DataSource object class implements the DataSource interface for non-pooling in addition to ConnectionPoolDataSource for pooling. To use non-pooled connections, modify the example in [“Creating a DataDirect SequeLink® Data Source Object” on page 542](#) so that it registers the SequeLink Data Source using the JNDI entry

```
<jdbc/SparkyOracle>
```

You can then run the TestDataSourceApp without any modification:

```
java TestDataSourceApp
```

Closing the Connection Pool

To ensure that the connection pool is closed correctly when an application stops running, the application must notify the DataDirect Connection Pool Manager when it stops. If your application runs on JRE 1.3 or higher, notification occurs automatically, so the application does not have to send notification. For JRE 1.2, the application must explicitly notify the pool manager of termination using a special close method defined by DataDirect as shown in the following example:

```
if (ds instanceof com.ddtek.pool.PooledConnectionDataSource) {  
    com.ddtek.pool.PooledConnectionDataSource pcds =  
(com.ddtek.pool.PooledConnectionDataSource) ds;  
    pcds.close();  
}
```

E Troubleshooting Using DataDirect Spy™

DataDirect Spy logs detailed information about calls issued by a running application to the JDBC driver. This information can help you troubleshoot problems when they occur.

Generating a DataDirect Spy™ Log

When you enable DataDirect Spy for a connection, you can customize DataDirect Spy logging for your needs by setting one or multiple options for DataDirect Spy. See [Chapter 7 “Tracking JDBC Calls” on page 315](#) for information about using DataDirect Spy and instructions on enabling and customizing logging.

Turning On and Off DataDirect Spy™ Logging

Once DataDirect Spy logging is enabled for a connection, you can turn on and off the logging at runtime using the `setEnableLogging` method in the `com.ddtek.jdbc.extensions.ExtLogControl` interface. When DataDirect Spy logging is enabled, all Connection objects returned to an application provide an implementation of the `ExtLogControl` interface.

For example, the following code turns off logging using `setEnabledLogging(false)`:

```
import com.ddtek.jdbc.extensions.*

// Get Database Connection
Connection con = DriverManager.getConnection
    ("jdbc:sequelink://QANT:4003;User=TEST;Password=secret;
    SpyAttributes=(log=(file)/tmp/spy.log");

((ExtLogControl) con).setEnabledLogging(false);
...
```

The `setEnabledLogging` method only turns on and off logging if DataDirect Spy logging has already been enabled for a connection; it does not set or change DataDirect Spy attributes. See [“Enabling DataDirect Spy™” on page 316](#) for information about enabling and customizing DataDirect Spy logging.

ExtLogControl Class

ExtLogControl Class	
Methods	Description
<code>void setEnabledLogging (boolean)</code>	If DataDirect Spy was enabled when the connection was created, you can turn on or off DataDirect Spy logging at runtime using this method. If true, logging is turned on. If false, logging is turned off. If DataDirect Spy logging was not enabled when the connection was created, calling this method has no effect.
<code>boolean getEnabledLogging ()</code>	Indicates whether DataDirect Spy logging was enabled when the connection was created and whether logging is turned on. If the returned value is true, logging is turned on. If the returned value is false, logging is turned off.

DataDirect Spy™ Log Example

The following example shows a DataDirect Spy log. The numbers in bold superscript are note indicators that correspond to the notes following the example. They provide explanations for the referenced text to help you understand the content of your own DataDirect Spy logs.

All rights reserved.¹

```
registerDriver:driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]2
```

```
*Driver.connect(jdbc:spy:{jdbc:sequelink://QANT:4003;databaseName=Oracle;})
    trying driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]3
```

```
spy>> Driver.connect(String url, Properties info)
spy>> url = jdbc:spy:{jdbc:sequelink://QANT:4003;databaseName=Oracle;
OSUser=qausser;OSPassword=null12}
spy>> info = {password=tiger, user=scott}
spy>> OK (Connection[1])4
```

```
getConnection returning driver[className=com.ddtek.jdbcspy.SpyDriver,
context=null,com.ddtek.jdbcspy.SpyDriver@1ec49f]5
```

```
spy>> Connection[1].getWarnings()
spy>> OK6
```

```
spy>> Connection[1].createStatement
spy>> OK (Statement[1])7
```

```
spy>> Statement[1].executeQuery(String sql)
spy>> sql = select empno,ename,job from emp where empno=7369
spy>> OK (ResultSet[1])8
```

```
spy>> ResultSet[1].getMetaData()
spy>> OK (ResultSetMetaData[1])9
```

```
spy>> ResultSetMetaData[1].getColumnCount()
spy>> OK (3)10
```

```
spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 1
spy>> OK (EMPNO)11

spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 2
spy>> OK (ENAME)12

spy>> ResultSetMetaData[1].getColumnLabel(int column)
spy>> column = 3
spy>> OK (JOB)13

spy>> ResultSet[1].next()
spy>> OK (true)14

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 1
spy>> OK (7369)15

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 2
spy>> OK (SMITH)16

spy>> ResultSet[1].getString(int columnIndex)
spy>> columnIndex = 3
spy>> OK (CLERK)17

spy>> ResultSet[1].next()
spy>> OK (false)18

spy>> ResultSet[1].close()
spy>> OK19

spy>> Connection[1].close()
spy>> OK20
```

NOTES:

- 1:** The Spy driver is registered. The spy>> prefix indicates that this line has been logged by Spy.
- 2:** The JDBC Driver Manager logs a message each time a JDBC driver is registered.
- 3:** This is the logging of the JDBC Driver Manager. It logs a message each time a JDBC application makes a connection.
- 4:** The application connects with the specified URL. The User Name and Password are specified using properties.
- 5:** This is the logging of the JDBC Driver Manager. It logs a message each time a successful connection is made.
- 6:** The application checks to see if there are any warnings. In this example, no warnings are present.
- 7 and 8:** The statement "select empno,ename,job from emp where empno=7369" is created.
- 9, 10, 11, 12, and 13:** Some metadata is requested.
- 14, 15, 16, and 17:** The first row is fetched.
- 18:** The application attempts to fetch the second row, but the database returned only one row for this query.
- 19:** After fetching all data, the result set is closed.
- 20:** The application finishes and disconnects.

F Developing ODBC Applications for Internationalization

This appendix provides an overview of how to design your applications for internationalization. Refer to the *SequeLink Administrator's Guide* for information about internationalization, localization, and Unicode.

Software that has been designed for internationalization can manage different linguistic and cultural conventions transparently and without additional modification. Software that has been designed for localization includes language translation, cultural data, and other components for meeting regional market requirements.

Although Unicode was developed to expand the number of available characters and ultimately to simplify data access in a world-wide setting, these goals have not been fully realized. The character set has been expanded, but data access still involves a number of conversions. This is because Unicode must be able to work with existing ANSI applications and because database vendors make data available in a number of different Unicode encoding formats, including UCS-2, UTF-16, and UTF-8.

ODBC drivers and the ODBC Driver Manager are the components responsible for processing function call and data encoding conversions. Developers of these components must code them to be able to recognize the type of function call and the various Unicode encoding schemes, and to make the appropriate conversions. The drivers and Driver Manager must make these conversions; Unicode data in a database can be accessed only by W function calls, and ANSI data can only be accessed by standard, non-W function calls.

Application developers, on the other hand, need only consider whether a Unicode or ANSI application is most appropriate for a particular circumstance and code their function calls appropriately—W function calls, such as `SQLConnectW`, for Unicode, or standard function calls, such as `SQLConnect`, for ANSI. They can also code an application to switch dynamically between Unicode and ANSI calls.

As Unicode applications and data become more prevalent, and more agreements are reached concerning encoding and implementation of Unicode, data access will become more efficient as the need for function call and data conversion is reduced.

References:

Java Internationalization: An Overview, John O'Connor,
<http://developer.java.sun.com/developer/technicalArticles/Intl/IntlIntro/>

Unicode Support in the Solaris Operating Environment, May 2000, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303-4900

Unicode and Non-Unicode ODBC Drivers

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

Function Calls

Instead of the standard ANSI SQL function calls, such as `SQLConnect`, Unicode applications employ W (wide) function calls, such as `SQLConnectW`. If the driver is a true Unicode driver, it can understand the "W" function calls and the Driver Manager can pass them through to the driver without conversion to ANSI. SequeLink Client *for* ODBC driver is a Unicode driver.

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

The following examples illustrate the conversion streams. The Driver Manager on Linux and UNIX prior to SequeLink Client 6.0 *for* ODBC assumes Unicode applications and Unicode drivers that use the same encoding (UTF-8). For SequeLink Client 6.0 *for* ODBC on Linux/UNIX, the Driver Manager determines the type of Unicode encoding of both the application and the driver, and performs conversions when the application and driver each use different types of encoding. This determination is made by checking two ODBC Environment Attributes: `SQL_ATTR_APP_UNICODE_TYPE` and `SQL_ATTR_DRIVER_UNICODE_TYPE`.

Unicode Application with the SequeLink ODBC Driver

An operation involving a Unicode application and the SequeLink Unicode driver is more efficient because no function conversion is involved. If the application and the driver use different types of encoding, there is some conversion overhead.

Windows

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

Linux/UNIX: SequeLink® Client 5.5 for ODBC

- 1 The Unicode application sends UTF-8 function calls to the Driver Manager.
- 2 The Driver Manager does not have to convert the UTF-8 function calls to ANSI. It passes the Unicode function call to the SequeLink *for* ODBC driver.
- 3 The driver returns UTF-8 argument values to the Driver Manager.
- 4 The Driver Manager returns UTF-8 function calls to the application.

Linux/UNIX: SequeLink® Client 6.0 for ODBC

- 1 The Unicode application sends function calls to the Driver Manager. The Driver Manager expects these function calls to be UTF-8 or UTF-16 based on the value of the SQL_ATTR_APP_UNICODE_TYPE attribute.
- 2 The Driver Manager passes the Unicode function call to the SequeLink *for* ODBC driver. The Driver Manager must perform function call conversions if the SQL_ATTR_APP_UNICODE_TYPE is different from SQL_ATTR_DRIVER_UNICODE_TYPE.
- 3 The driver returns argument values to the Driver Manager. Whether these are UTF-8 or UTF-16 argument values is based on the value of the SQL_ATTR_DRIVER_UNICODE_TYPE attribute.
- 4 The Driver Manager returns appropriate function calls to the application, based on the SQL_ATTR_DRIVER_UNICODE_TYPE attribute. The Driver Manager must perform function call conversions if the SQL_ATTR_APP_UNICODE_TYPE is different from SQL_ATTR_DRIVER_UNICODE_TYPE.

Non-Unicode Application with the SequeLink® ODBC Driver

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

Windows

- 1 The non-Unicode application sends ANSI function calls to the Driver Manager.
- 2 The Driver Manager converts the ANSI function calls to UCS-2/UTF-16. It passes the Unicode function calls with

UCS-2/UTF-16 argument values to the SequeLink *for* ODBC driver.

- 3 The SequeLink driver returns UCS-2/UTF-16 argument values to the Driver Manager.
- 4 The Driver Manager converts the UCS-2/UTF-16 argument values to ANSI and returns these argument values to the application.

Linux/UNIX: SequeLink® Client for ODBC 5.5

- 1 The non-Unicode application sends ANSI function calls to the Driver Manager.
- 2 The Driver Manager converts the ANSI function calls to UTF-8. It passes the Unicode function calls with UTF-8 argument values to the SequeLink *for* ODBC driver.
- 3 The SequeLink ODBC driver returns UTF-8 argument values to the Driver Manager.
- 4 The Driver Manager converts the UTF-8 argument values to ANSI and returns them to the application.

Linux/UNIX: SequeLink® Client for ODBC 6.0

- 1 The non-Unicode application sends ANSI function calls to the Driver Manager. The Driver Manager expects these function calls to be UTF-8 or UTF-16 based on the value of the SQL_ATTR_APP_UNICODE_TYPE attribute. Because non-Unicode applications will not set this attribute, the default of UTF-8 is used for the conversion.
- 2 The Driver Manager converts the ANSI function calls to UTF-8. It passes the Unicode function calls with UTF-8 argument values to the SequeLink ODBC driver.

- 3 The SequeLink *for* ODBC driver returns argument values to the Driver Manager. The value of the `SQL_ATTR_APP_UNICODE_TYPE` attribute determines whether these argument values are UTF-8 or UTF-16.
- 4 The Driver Manager converts the UTF-8 or UTF-16 argument values to ANSI and returns them to the application.

Data

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

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

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

capable of converting SQL_C_CHAR to SQL_WCHAR, and vice versa.

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

When there is a difference in types of data, it must be converted from one type to another at the driver or Driver Manager level, which involves additional overhead.

The Unicode SequeLink driver, NOT the Driver Manager, converts SQL_C_CHAR (ANSI) data to SQL_WCHAR (Unicode) data, and vice versa, as well as SQL_C_WCHAR (Unicode) data to SQL_CHAR (ANSI) data, and vice versa.

The driver must use client code page information (Active Code Page on Windows, IANAAppCodePage attribute on Linux/UNIX) to determine which ANSI codepage to use for the conversions.

Developing ODBC Applications on Linux/UNIX

It is important to remember some differences in developing applications on Linux/UNIX:

- ["Using Double-Byte Character Sets on Linux/UNIX" on page 565](#)
- ["Using UTF-16 for your Applications on Linux/UNIX" on page 566](#)

Using Double-Byte Character Sets on Linux/UNIX

The *SequeLink* for ODBC UNIX drivers can use double-byte character sets. The drivers normally use the character set defined by the default locale C unless explicitly pointed to another character set. The default locale C corresponds to the 7-bit USASCII character set.

Use the following procedure to define a different character set for the locale:

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

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

The `setlocale(LC_ALL, "")` function selects the program's entire locale and may be used to change the program's entire local. The "" corresponds to the value of the associated environment variables, `LC_*` and `LANG`. If this line is not present, the default locale C is used. If this line is present and if `LANG` or `LC_TYPE` is set, the locale character set is determined based on these values. `LC_TYPE` overwrites the `LANG` setting.

If LANG and LC_TYPE is either not set or is set to NULL, the default locale C is used.

- 2 Set the LC_CTYPE and/or LANG environment variable to the appropriate character set. The Linux/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 SequeLink Client 6.0 for ODBC ships the utility `ivslkcheckcp` to check the following information:

- What value to choose for the IANAAppCodePage
- What code page will be used by your application

Using UTF-16 for your Applications on Linux/UNIX

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

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

- The definition of `SQLWCHAR` in the ODBC header files must be switched from `char *` to `short *`. To do this, the application uses `#define SQLWCHARSHORT`.
- The application must set the ODBC Environment Attribute `SQL_ATTR_APP_UNICODE_TYPE` to a value of `SQL_DD_CP_UTF16`.

The Driver Manager on Linux/UNIX

The Driver Manager on Linux/UNIX shipped with DataDirect SequeLink *for* ODBC 6.0 determines the type of Unicode encoding of both the application and the driver, and performs conversions when the application and driver each use different types of encoding.

Unicode ODBC drivers on Linux/UNIX can be written with either UTF-8 or UTF-16 encoding. This would normally mean that a UTF-8 application could not work with a UTF-16 driver, and, conversely, that a UTF-16 application could not work with a UTF-8 driver. To accomplish the goal of being able to use a single UTF-8 or UTF-16 application with either a UTF-8 or UTF-16 driver, the Driver Manager must be able to determine with which type of encoding the application and driver are written and, if necessary, convert them accordingly.

To make this determination, the Driver Manager supports two ODBC Environment Attributes: `SQL_ATTR_APP_UNICODE_TYPE` and `SQL_ATTR_DRIVER_UNICODE_TYPE`, each with possible values of `SQL_DD_CP_UTF8` and `SQL_DD_CP_UTF16`. The default value is `SQL_DD_CP_UTF8`.

There are several steps the Driver Manager must undertake before actually connecting to the driver to achieve this goal.

- 1 Determine the application Unicode type: Applications that want to use UTF-16 encoding for their string types need to set `SQL_ATTR_APP_UNICODE_TYPE` accordingly before connecting to any driver. When the Driver Manager reads this attribute, it expects all string arguments to the ODBC W functions to be in the specified Unicode. This attribute also indicates how the `SQL_C_WCHAR` buffers must be encoded.

- 2 Determine the driver Unicode type: The Driver Manager must determine through which Unicode encoding the driver supports its "W" functions. This is done as follows:
 - `SQLGetEnvAttr(SQL_ATTR_DRIVER_UNICODE_TYPE)` is called in the driver by the Driver Manager. The driver, if capable, returns either `SQL_DD_CP_UTF16` or `SQL_DD_CP_UTF8` to indicate to the Driver Manager which encoding it expects.
 - If the preceding call to `SQLGetEnvAttr` fails, the Driver Manager looks either in the Data Source section of the `odbc.ini` specified by the connection string or in the connection string itself for a connection option called `DriverUnicodeType`. The valid values for this option are 1 (UTF-16) or 2 (UTF-8). The Driver Manager assumes that the Unicode encoding of the driver corresponds to the value specified.
 - If neither of the preceding attempts are successful, the Driver Manager assumes that the Unicode encoding of the driver is UTF-8.
- 3 Determine if the driver supports `SQL_ATTR_WCHAR_TYPE`: `SQLSetConnectAttr(SQL_ATTR_WCHAR_TYPE, X)` is called in the driver by the Driver Manager, where X is either `SQL_DD_CP_UTF8` or `SQL_DD_CP_UTF16`, depending on the value of the `SQL_ATTR_APP_UNICODE_TYPE` environment setting. If the driver returns any error on this call to `SQLSetConnectAttr`, the Driver Manager assumes that the driver does not support this connection attribute.

In the case of an error, the Driver Manager converts all data bound as `SQL_C_WCHAR` to the application Unicode type as specified by `SQL_ATTR_APP_UNICODE_TYPE`. The Driver Manager also converts all bound parameter data from the application Unicode type to the driver Unicode type specified by `SQL_ATTR_DRIVER_UNICODE_TYPE`.

Based on the information it has gathered prior to connection, the Driver Manager either does not have to convert function calls, or it converts to either UTF-8 or UTF-16 all string arguments to calls to the ODBC W functions before calling the driver.

Values for IANAAppCodePage Connection String Attribute



[Table F-1](#) lists supported values, along with a description, for the IANAAppCodePage connection string attribute at the time of this publication. Support for additional values may have been added since publication time; therefore, for up-to-date values, go to:

<http://www.datadirect.com/support/troubleshooting/su-faq-iana/index.ssp>

See [“Connecting Using a Connection String” on page 57](#) for more information about this connection string attribute.

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

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

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

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

- 3 Check the table in this appendix to make sure that the MIBenum value you looked up on the IANA Web page is supported by SequeLink Client *for* ODBC. If the value is not listed, contact DataDirect Technologies technical support to request that support for that value be added.

Table F-1. IANAAppCodePage Values

IANAAppCodePage	IANA Character Set Name
3	US_ASCII
4	ISO_8859_1
5	ISO_8859_2
6	ISO_8859_3
8	ISO_8859_5
9	ISO_8859_6
10	ISO_8859_7
11	ISO_8859_8
12	ISO_8859_9
17	Shift_JIS
18	EUC_JP
38	EUC_KR
106	UTF-8
109	ISO_8859_13
111	ISO_8859_15
113	GBK
114	GB18030
2004	HP_ROMAN8
2009	IBM850
2025	2025 GB2312
2026	Big5
2084	KOI8_R
2088	KOI8-U

Table F-1. IANAAppCodePage Values (cont.)

IANAAppCodePage	IANA Character Set Name
2251	WINDOWS_1251
2252	WINDOWS_1252
2259	TIS_620
10001	IBM-856
10003	IBM-921
10004	IBM-922
10012	IBM-943
10024	IBM-1046
10030	IBM-1124

Solaris

[Table F-2](#) lists supported code pages, IANA character set name and IANAAppCodePage for the SequeLink Client 6.0 or higher for ODBC on the Solaris platform.

Table F-2. Code Pages Supported on Solaris

Code page	IANA Character Set Name	IANAAppCodePage
646	US_ASCII	3
ISO8859-1	ISO_8859_1	4
ISO8859-2	ISO_8859_2	5
ISO8859-5	ISO_8859_5	8
ISO8859-6	ISO_8859_6	9
ISO8859-7	ISO_8859_7	10
ISO8859-8	ISO_8859_8	11

Table F-2. Code Pages Supported on Solaris *(cont.)*

Code page	IANA Character Set Name	IANAAppCodePage
ISO8859-9	ISO_8859_9	12
PCK	Shift_JIS	17
eucJP	EUC_JP	18
ISO8859-13	ISO_8859_13	109
ISO8859-15	ISO_8859_15	111
GBK	GBK	113
UTF-8	UTF-8	106
gb2312	GB2312	2025
BIG5	Big5	2026
KOI8-R	KOI8_R	2084
TIS620.2533	TIS_620	2259

HP

Table F-3 lists supported code pages, IANA character set name and IANAAppCodePage for the SequeLink Client 6.0 or higher for ODBC on the HP-UX platform.

Table F-3. Code Pages Supported on HP-UX

Code page	IANA Character Set Name	IANAAppCodePage
iso88591	ISO_8859_1	4
iso88592	ISO_8859_2	5
iso88595	ISO_8859_5	8
iso88596	ISO_8859_6	9
iso88597	ISO_8859_7	10

Table F-3. Code Pages Supported on HP-UX (cont.)

Code page	IANA Character Set Name	IANAAppCodePage
iso88598	ISO_8859_8	11
iso88599	ISO_8859_9	12
utf8	UTF-8	106
big5	Big5	2026
tis620	TIS_620	2259
roman8	HP_ROMAN8	2004

AIX

[Table F-4](#) lists supported code pages, IANA character set name and IANAAppCodePage for the SequeLink Client 6.0 or higher for ODBC on the AIX platform.

Table F-4. Code Pages Supported on AIX

Code Page	IANA Character Set Name	IANAAppCodePage
kIACS_MIB_US_ASCII	US_ASCII	3
ISO8859-1	ISO_8859_1	4
ISO8859-2	ISO_8859_2	5
ISO8859-5	ISO_8859_5	8
ISO8859-6	ISO_8859_6	9
ISO8859-7	ISO_8859_7	10
ISO8859-8	ISO_8859_8	11
ISO8859-9	ISO_8859_9	12
IBM-eucJP	EUC_JP	18
IBM-eucKR	EUC_KR	38

Table F-4. Code Pages Supported on AIX (cont.)

Code Page	IANA Character Set Name	IANAAppCodePage
ISO8859-15	ISO_8859_15	111
GBK	GBK	113
UTF-8	UTF-8	106
IBM-850	IBM-850	2009
big5	Big5	2026
IBM-1252	WINDOWS_1252	2252
TIS-620	TIS_620	2259
IBM-856	IBM-856	10001
IBM-921	IBM-921	10003
IBM-922	IBM-922	10004
IBM-943	IBM-943	10012
IBM-1046	IBM-1046	10024
IBM-1124	IBM-1124	10030

Linux

Table F-5 lists supported code pages, IANA character set name and IANAAppCodePage for the SequeLink Client 6.0 or higher for ODBC on the Linux platform.

Table F-5. Code Pages Supported on Linux

Code Page	IANA Character Set Name	IANAAppCodePage
ANSI_X3.4-1968	US_ASCII	3
ISO-8859-1	ISO_8859_1	4
ISO-8859-2	ISO_8859_2	5
ISO-8859-3	ISO_8859_3	6

Table F-5. Code Pages Supported on Linux (cont.)

Code Page	IANA Character Set Name	IANAAppCodePage
ISO-8859-5	ISO_8859_5	8
ISO-8859-5	ISO_8859_6	9
ISO-8859-7	ISO_8859_7	10
ISO-8859-8	ISO_8859_8	11
ISO-8859-9	ISO_8859_9	12
EUC-JP	EUC_JP	18
EUC-KR	EUC_KR	38
ISO-8859-13	ISO_8859_13	109
ISO-8859-15	ISO_8859_15	111
GBK	GBK	113
GB18030	GB18030	114
UTF-8	UTF-8	106
GB2312	2025 GB2312	2025
BIG5	Big5	2026
KOI8-R	KOI8_R	2084
KOI8-U	KOI8-U	2088
CP1251	WINDOWS_1251	2251
TIS-620	TIS_620	2259

Windows

Table F-6 lists supported code pages, code page name, MIB number, and IANA character set name for the SequeLink Client 6.0 or higher *for* ODBC on the Windows platform.

Table F-6. Code Pages Supported on Windows

Codepage	Codepage Name	MIBenum	IANA Character Set Name
932	Japanese Shift-JIS	17	Shift_JIS
936	Simplified Chinese GBK	113	GBK
949	Korean	10015	WINDOWS_949
950	Simplified Chinese Big5	2026	Big5
1250	Central Europe	2250	WINDOWS_1251
1251	Cyrillic	2251	WINDOWS_1251
1252	Latin I	2252	WINDOWS_1252
1253	Greek	2253	WINDOWS_1253
1254	Turkish	2254	WINDOWS_1254
1255	Hebrew	2255	WINDOWS_1255
1256	Arabic	2256	WINDOWS_1256
1257	Baltic	2257	WINDOWS_1257
1258	Vietnam	2258	WINDOWS_1258

G .NET Code Examples

This appendix includes code examples of typical database access tasks in ADO.NET using the Microsoft .NET Framework 1.x. All of the examples are written in C#.

You must modify the connection strings in the examples to work in your environment. All of the examples include calls to `MessageBox`, which is in the `System.Windows.Forms` namespace.

Sample Tables Used in the Code Examples

Many of the samples in this appendix use the `emp` and `dept` tables.

- To create the tables on Oracle, use the script in [“Sample Tables for Oracle” on page 578](#).
- To create the tables on Sybase, use the script in [“Sample Tables for Sybase” on page 579](#).

Sample Tables for Oracle

The following script can be run in SQL*Plus. Refer to the Oracle documentation for details.

```
CREATE TABLE emp (
    empno      NUMBER(4) NOT NULL PRIMARY KEY,
    ename       VARCHAR2(10),
    job         VARCHAR2(9),
    mgr         NUMBER(4),
    hiredate    DATE,
    sal         NUMBER(7,2),
    comm        NUMBER(7,2),
    dept        NUMBER(2));

alter session set NLS_DATE_FORMAT = 'MM-DD-YYYY';

begin
insert into emp values (1,'JOHNSON','ADMIN',6,'12-17-1990',18000,NULL,4);
insert into emp values (2,'HARDING','MANAGER',9,'02-02-1998',50000,300,3);
insert into emp values (3,'TAFT','SALESMAN',2,'01-02-1996',25000,500,3);
insert into emp values (4,'HOOVER','SALESMAN',2,'04-02-1990',25000,NULL,3);
insert into emp values (5,'LINCOLN','TECH',6,'06-23-1994',22500,1400,4);
insert into emp values (6,'GARFIELD','MANAGER',9,'05-01-1993',50000,NULL,4);
insert into emp values (7,'HARRISON','TECH',6,'09-22-1997',25000,NULL,4);
insert into emp values (8,'GRANT','ENGINEER',10,'03-30-1997',30000,NULL,2);
insert into emp values (9,'JACKSON','CEO',NULL,'01-01-1990',75000,NULL,4);
insert into emp values (10,'FILLMORE','MANAGER',9,'08-09-1994',50000,NULL,2);
insert into emp values (11,'ADAMS','ENGINEER',10,'03-15-1996',30000,NULL,2);
insert into emp values (12,'WASHINGTON','ADMIN',6,'04-16-1998',18000,NULL,4);
insert into emp values (13,'MONROE','ENGINEER',10,'12-03-2000',30000,NULL,2);
insert into emp values (14,'ROOSEVELT','CPA',9,'10-12-1995',35000,NULL,1);
end;
/

CREATE TABLE dept (
    deptno NUMBER(2) NOT NULL,
    dname  VARCHAR2(14),
    loc    VARCHAR2(13));
```

```

begin
insert into dept values (1,'ACCOUNTING','ST LOUIS');
insert into dept values (2,'RESEARCH','NEW YORK');
insert into dept values (3,'SALES','ATLANTA');
insert into dept values (4, 'OPERATIONS','SEATTLE');
end;
/

```

Sample Tables for Sybase

The following script can be run in ISQL. Refer to the Sybase documentation for details.

```

CREATE TABLE emp (
    empno      INT PRIMARY KEY,
    ename      VARCHAR(10),
    job        VARCHAR(9),
    mgr        INT NULL,
    hiredate   DATETIME,
    sal        NUMERIC(7,2),
    comm       NUMERIC(7,2) NULL,
    dept       INT)

go

begin
insert into emp values (1,'JOHNSON','ADMIN',6,'12-17-1990',18000,NULL,4)
insert into emp values (2,'HARDING','MANAGER',9,'02-02-1998',50000,300,3)
insert into emp values (3,'TAFT','SALESMAN',2,'01-02-1996',25000,500,3)
insert into emp values (4,'HOOVER','SALESMAN',2,'04-02-1990',25000,NULL,3)
insert into emp values (5,'LINCOLN','TECH',6,'06-23-1994',22500,1400,4)
insert into emp values (6,'GARFIELD','MANAGER',9,'05-01-1993',50000,NULL,4)
insert into emp values (7,'POLK','TECH',6,'09-22-1997',25000,NULL,4)
insert into emp values (8,'GRANT','ENGINEER',10,'03-30-1997',30000,NULL,2)
insert into emp values (9,'JACKSON','CEO',NULL,'01-01-1990',75000,NULL,4)
insert into emp values (10,'FILLMORE','MANAGER',9,'08-09-1994',50000,NULL,2)
insert into emp values (11,'ADAMS','ENGINEER',10,'03-15-1996',30000,NULL,2)
insert into emp values (12,'WASHINGTON','ADMIN',6,'04-16-1998',18000,NULL,4)

```

```
insert into emp values (13,'MONROE','ENGINEER',10,'12-03-2000',30000,NULL,2)
insert into emp values (14,'ROOSEVELT','CPA',9,'10-12-1995',35000,NULL,1)
end

go

CREATE TABLE dept (
    deptno INT NOT NULL,
    dname  VARCHAR(14),
    loc    VARCHAR(13))
go

begin
insert into dept values (1,'ACCOUNTING','ST LOUIS')
insert into dept values (2,'RESEARCH','NEW YORK')
insert into dept values (3,'SALES','ATLANTA')
insert into dept values (4,'OPERATIONS','SEATTLE')
end

go
```

Retrieving Data Using a DataReader

The DataReader provides the fastest but least flexible way to retrieve data from the database. Data is returned as a read-only, forward-only stream of data that is returned one record at a time. If you need to retrieve many records rapidly, using a DataReader would require less memory than a DataSet, which would need to create a large table to hold the results.

The following code example shows how to execute a simple query on a Sybase database and read the results using a DataReader.

NOTE: The example requires the emp table (see [“Sample Tables for Sybase” on page 579](#)). The Sybase database in this example does not require the Database connection string option.

```

// Open connection to SequeLink Server on Sybase database
SequeLinkConnection Conn;
Conn = new SequeLinkConnection("host=bowhead;port=19996;User ID=test01;
                                Password=test01");

Conn.Open();

// Create a SQL command
string strSQL = "SELECT ename FROM emp WHERE sal>50000";
SequeLinkCommand myCommand = new SequeLinkCommand(strSQL, Conn);
SequeLinkDataReader myDataReader;
myDataReader = myCommand.ExecuteReader()

try
{
    while (myDataReader.Read())
    {
        Console.WriteLine("High salaries:" + my DataReader["ename"].ToString());
    }
}

catch (Exception ex)

{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}

// Always close the DataReader
myDataReader.Close();

// Close the connection
Conn.Close();

```

Using a Local Transaction With a DataReader

The following example shows how to use a local transaction with a `SequeLinkDataReader` object on the SequeLink server Sparky on Oracle.

NOTE: The sample uses the emp table (see [“Sample Tables for Oracle” on page 578](#)). The Oracle database in this example does not require the Database connection string option.

```
SequeLinkConnection  DBConn;
DBConn = new SequeLinkConnection("host=norman;Port=19996;User ID=test01;
                                Password=test01");
SequeLinkCommand      DBCmd = new SequeLinkCommand();
SequeLinkTransaction DBTxn = null;
try
{
    DBConn.Open();
    DBTxn = DBConn.BeginTransaction();
    // Set the Connection property of the Command object
    DBCmd.Connection = DBConn;
    // Set the text of the Command to the INSERT statement
    DBCmd.CommandText = "INSERT INTO emp VALUES (15,'HAYES','ADMIN',6,
                                '17-APR-2007',18000,NULL,4)";
    // Set the transaction property of the Command object
    DBCmd.Transaction = DBTxn;
    // Execute the statement
    DBCmd.ExecuteNonQuery();
    // Now commit the transaction
    DBTxn.Commit();
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show(ex.Message);
    // If anything failed after the connection was opened, roll back the
    // transaction
}
```

```

    if (DBTxn != null)
    {
        DBTxn.Rollback();
    }
}
// Close the connection
DBConn.Close();

```

Using a Distributed Transaction

The following code example, which uses the emp table (see [“Sample Tables for Oracle” on page 578](#)), shows how to use a distributed transaction to connect to two different Oracle servers.

NOTE: Microsoft Distributed Transaction Coordinator must be running on all clients and servers. The Oracle databases in this example do not require the Database connection string option.

```

using System;
using System.EnterpriseServices;
using DDTek.SequeLink;

namespace DistributedTransaction
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    public class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args) {

            SequeLinkConnection DBConn1;

```

```

        DBConn1 = new SequeLinkConnection("host=norman;SID=test;Port=1521;
                                           User ID=test01;Password=test01;Enlist=true");
        SequeLinkConnection DBConn2;
        DBConn2 = new SequeLinkConnection("host=Carrie;SID=test;Port=1521;
                                           User ID=test01;Password=test01;Enlist=true");
        try {
            DBConn1.Open();
            DBConn2.Open();

            DistributedTran myDistributedTran = new DistributedTran();
            myDistributedTran.TestDistributedTransaction(DBConn1, DBConn2);

            // Note, the connections used in distributed transaction must
            // only be closed outside of transaction methods, because after
            // these methods finish, DTC still needs these
            // connections to complete commit/rollback commands.
            DBConn1.Close();
            DBConn2.Close();
        }
        catch (Exception e) {
            System.Console.WriteLine("Error returned: " + e.Message);
        }
    }
}

/// <summary>
/// To use distributed transactions in .NET, we need a ServicedComponent
/// derived class with transaction attribute declared as "Required".
/// </summary>
[Transaction(TransactionOption.Required) ]
public class DistributedTran : ServicedComponent {

    /// <summary>
    /// This method executes two SQL statements, one on each database
    /// server. If both are successful, both are committed by DTC after the
    /// method finishes. However, if an exception is thrown, both will be
    /// rolled back by DTC.
    /// </summary>
    [AutoComplete]

```



```

public void TestDistributedTransaction(
    SequeLinkConnection DBConn1,
    SequeLinkConnection DBConn2) {

    // The following Insert statement goes to the first server, norman.
    // This Insert statement does not produce any errors.
    String DBCmdSql1 = "INSERT INTO emp VALUES
                        (15, 'HAYES', 'ADMIN', 6, '17-NOV-2006', 18000, NULL, 4)";

    // The following Delete statement goes to the second server,
    // Carrie. Because the Raises table does not exist on Carrie,
    // the code throws an exception.
    String DBCmdSql2 = "DELETE * FROM Raises WHERE sal > 100000";

    SequeLinkCommand DBCmd1 = new SequeLinkCommand(DBCmdSql1, DBConn1);
    SequeLinkCommand DBCmd2 = new SequeLinkCommand(DBCmdSql2, DBConn2);

    DBCmd1.ExecuteNonQuery();

    // This command results in an exception, which automatically rolls
    // back the DBCmd1 command on the other server.
    DBCmd2.ExecuteNonQuery();
}
}

```

Using the CommandBuilder

The following example uses the CommandBuilder to create a SQL statement on the SequeLink Server Sparky on Oracle.

NOTE: This example uses the "emp" sample table (see ["Sample Tables for Oracle" on page 578](#)). The Oracle database in this example does not require the Database connection string option.

```
SequeLinkConnection    DBConn;
DBConn = new SequeLinkConnection("host=norman;Port=19996;User ID=test01;
                                Password=test01");

SequeLinkDataAdapter    myDataAdapter = new SequeLinkDataAdapter();
SequeLinkCommand    DBCmd = new SequeLinkCommand("SELECT * FROM EMP",DBConn);
myDataAdapter.SelectCommand = DBCmd;
// Set up the CommandBuilder
SequeLinkCommandBuilder
    CommBuild = new SequeLinkCommandBuilder(myDataAdapter);
DataSet    myDataSet = new DataSet();
try
{
    DBConn.Open();
    myDataAdapter.Fill(myDataSet);
    // Now change the salary of the first employee
    DataRow    myRow;
    myRow = myDataSet.Tables["Table"].Rows[0];
    myRow["sal"] = 95000;
    // Tell the DataAdapter to resynch with the Oracle server.
    // Without the CommandBuilder, this line would fail.
    myDataAdapter.Update(myDataSet);
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
DBConn.Close();
```

Retrieving a Result Set Using a DataAdapter Object

The DataAdapter is a bridge between the ADO.NET DataSet object and the underlying DBMS. The DataAdapter can be used to fill a DataSet with a result set or multiple result sets. Although the DataAdapter also includes functionality to propagate changes to the result set back to the DBMS, this example only shows how to set up the DataAdapter to retrieve the data and Fill a DataSet.

NOTE: The example requires the emp table (see [“Sample Tables for Sybase” on page 579](#)). The Sybase database in this example does not require the Database connection string option.

```
// Open connection to SequeLink Server on Sybase database
SequeLinkConnection Conn;
Conn = new SequeLinkConnection("host=bowhead;port=19996;User ID=test01;
                                Password=test01");

Conn.Open();

// Create a SQL command
string sqlSEL = "SELECT NAME FROM emp WHERE sal>'50000'";
SequeLinkDataAdapter myDataAdapter = new SequeLinkDataAdapter(sqlSEL,
Conn);

DataSet myDataSet = new DataSet("salDataSet");

try
{
    myDataAdapter.Fill(myDataSet, "emp");
}

catch(Exception ex) {
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
```

```
finally
// Close the connection
{
Conn.Close();
}
```

Limiting the Rows Returned by a Select Statement

The following example shows how to use a Command object to limit the number of rows returned by the Oracle server.

NOTE: The example requires the emp table (see [“Sample Tables for Oracle” on page 578](#)). The Oracle database in this example does not require the Database connection string option.

```
SequeLinkConnection DBConn;
DBConn = new SequeLinkConnection("host=baleen;Port=19996;User ID=test01;
                                Password=test01");

SequeLinkDataAdapter myDataAdapter = new SequeLinkDataAdapter();
SequeLinkCommand DBCmd = new SequeLinkCommand("SELECT * FROM EMP",DBConn);
myDataAdapter.SelectCommand = DBCmd;
DataSet myDataSet = new DataSet();
try
{
    DBConn.Open();
    // Set the RowsetSize on the Command object to limit the number
    // of rows returned
    DBCmd.RowsetSize = 10;
    myDataAdapter.SelectCommand = DBCmd;
    myDataAdapter.Fill(myDataSet);
    // Normally, emp contains 14 rows.
    // But with RowsetSize set to 10, the Command object
    // will limit the result set to 10 rows.

    MessageBox.Show(Convert.ToString(myDataSet.Tables["Table"].Rows.Count));
```

```

}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
DBConn.Close();

```

Updating Data in a DataSet

When updating a row at the data source, you call the `Update` statement. The `Update` statement uses parameters that contain the unique identifier (such as the primary key), and the columns to be updated, as shown in the following example:

```

[C#]
string updateSQL As String = "UPDATE emp SET sal = ?, job = ? +
    = WHERE empno = ?;

```

The parameterized query statements define the parameters that will be created. See ["Parameter Markers" on page 379](#) for more information about using parameters.

The following code example uses the `Parameters.Add` method to create the parameters for the preceding SQL statement, fill a `DataSet`, and print the updated table.

NOTE: The example requires the `emp` table (see ["Sample Tables for Oracle" on page 578](#)). The Oracle database in this example does not require the Database connection string option.

```

void test () {
    string updateSQL = "UPDATE emp SET sal = ?, job = ? WHERE empno = ?";
    string selectText = "SELECT sal, job, empno FROM emp";
    string updateText = "UPDATE emp SET sal = ?, job = ? WHERE empno = ?";
    SequeLinkConnection con = new SequeLinkConnection("host=baleen;
        User ID=test01;Password=test01");

```

```

SequeLinkDataAdapter adapter = new SequeLinkDataAdapter(selectText, Conn);

SequeLinkCommand updateCommand = new SequeLinkCommand(updateText, Conn);
updateCommand.Parameters.Add("@sal", SequeLinkDbType.Decimal, 0, "SAL");
updateCommand.Parameters.Add("@job", SequeLinkDbType.Varchar, 9, "JOB");
updateCommand.Parameters.Add("@empno", SequeLinkDbType.Int, 0,
                                "EMPNO");

updateCommand.Parameters["@empno"].SourceVersion = DataRowVersion.Original;
adapter.UpdateCommand = updateCommand;

DataSet myDataSet = new DataSet("emp");
adapter.Fill(myDataSet, "emp");

// print
PrintTable(myDataSet);

// Give employee number 11 a promotion and a raise
DataRow changeRow = myDataSet.Tables["emp"].Rows[10];
changeRow["sal"] = "35000";
changeRow["job"] = "MANAGER";

// Send back to database and reprint
try
{
    adapter.Update(myDataSet, "emp");
    myDataSet.Dispose();
    myDataSet = new DataSet();
    adapter.Fill(myDataSet, "emp");
    PrintTable(myDataSet);
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
Conn.Close();
}

```

Calling a Stored Procedure

You call stored procedures using a Command object. When you issue a command on a stored procedure, you must set the CommandType of the Command object to StoredProcedure.

The following code example shows how to execute a stored procedure on a Sybase database and read the results using a DataReader. The example uses the emp table (see [“Sample Tables for Sybase” on page 579](#)). The Sybase database in this example does not require the Database connection string option.

Create the following stored procedure:

```
CREATE PROCEDURE GetEmpSalary
    (@empno int,
     @sal char(7) output)
AS
SELECT @sal = sal FROM emp WHERE empno = @empno
```

The following code example executes the GetEmpSalary stored procedure:

```
// Open connection to SequeLink Server on Sybase database
SybaseConnection Conn;
Conn = new SequeLinkConnection("host=bowhead;port=19996;User ID=test01;
                                Password=test01");
Conn.Open();

// Make a command object for the stored procedure
// You must set the CommandType of the Command object
// to StoredProcedure
SequeLinkCommand DBCmd = new SequeLinkCommand("GetEmpSalary",Conn);
DBCmd.CommandType = CommandType.StoredProcedure;
SequeLinkDataReader myDataReader;
try
{
    myDataReader = myCommand.ExecuteReader()
    myDataReader.Close();
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
Conn.Close();
```

Retrieving Warning Information

The data providers handle database server warnings through the `InfoMessage` delegates on the `Connection` objects.

The following example shows how to retrieve a warning from a Sybase server:

```
// Define an event handler
public void myHandler(object sender, SequeLinkInfoMessageEventArgs e)
{
    // Display any warnings in a messagebox
    MessageBox.Show (e.Message, "This is a Warning.");
}
```

Add the following code to a method and call it:

```
SequeLinkConnection Conn;
Conn = new SequeLinkConnection("host=bowhead;port=19996;User ID=test01;
                                Password=test01");

SequeLinkCommand DBCmd = new SequeLinkCommand("print 'This is a warning.'",
                                                Conn);
SequeLinkDataReader myDataReader;
try
{
    Conn.InfoMessage += new SequeLinkInfoMessageEventHandler(myHandler);
    Conn.Open();
    myDataReader = DBCmd.ExecuteReader();
    // This will throw a SequeLinkInfoMessageEvent as the print
    // statement generates a warning.

}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
Conn.Close();
```

Retrieving a Scalar Value

You can use the `ExecuteScalar` method of the `Command` object to return a single value, such as a sum or a count, from the database. The `ExecuteScalar` method returns the value of the first column of the first row of the result set.

The following code example, which requires the `emp` table (see [“Sample Tables for Sybase” on page 579](#)), shows how to retrieve the count of a specified group:

```
// Retrieve the number of employees who make more than $50000
// from the emp table

// Open connection to SequeLink Server on Sybase database
SequeLinkConnection Conn;
Conn = new SequeLinkConnection("host=bowhead;port=19996;User ID=test01;
                               Password=test01");

Conn.Open();

// Make a command object
SequeLinkCommand salCmd = new SequeLinkCommand("SELECT COUNT(sal) FROM emp
                                                WHERE sal > '50000'", Conn);

try
{
    int count = (int)salCmd.ExecuteScalar();
}
catch (Exception ex)
{
    // Display any exceptions in a messagebox
    MessageBox.Show (ex.Message);
}
// Close the connection
Conn.Close();
```

Index

Symbols

.NET

- closing opened connections 400
- Framework data types for .NET data provider 390
- security 405, 406
- supported public objects 396

.NET Client

- assembly name 378
- configuring connection failover 375
 - Alternate Servers connection string option 370
 - Connection Retry Count connection string option 371
 - Connection Retry Delay connection string option 372
 - Load Balancing connection string option 373
- connecting to a database 383
- connection string attributes 370
- creating a connection pool 364
- data types 390
- data types supported
 - DB2 UDB on Linux/UNIX/Windows 460
 - DB2 UDB on z/OS 452, 455
 - Informix 462
 - Microsoft SQL Server 466
 - Oracle 473, 478
 - Sybase 480
- database errors 408
- event handling 396
- isolation levels 395
- mapping parameter types 392
- namespace 389

object

- InfoMessageEventArgs 402
- SequeLinkCommand 397
- SequeLinkCommandBuilder 398
- SequeLinkConnection 399
- SequeLinkDataAdapter 400
- SequeLinkDataReader 401
- SequeLinkError 401
- SequeLinkErrorCollection 401
- SequeLinkException 402
- SequeLinkTrace 403
- SequeLinkTransaction 405
- parameter arrays 379
- parameter markers 379
- provider-specific methods and properties supported 397
- SequeLink client errors 407
- setting a savepoint 405
- thread support 395
- transaction support 381
- ? for parameter markers 379

Numerics

- 64-bit ODBC Client (Linux/UNIX)
 - configuring 53
 - example 54
 - executing the shell script 55
- 64-bit platform support for JDBC Client 215

A

- ABS function 442
- Access Order 190, 203
- accessor support 177, 198
- ACOS function 442
- Active Sessions 194
- adding connections to a connection pool 365
- ADO
 - Command object 190
 - Connection object 193
 - mapping OLE DB methods 190
 - OLE DB interfaces supported 188
 - Recordset object 201
- ADO Client
 - data types supported
 - DB2 UDB on z/OS 453
- ADO client data sources
 - See client data sources
- ADO data provider
 - client load balancing 168
 - connection failover
 - configuring 164
 - connection properties
 - for connection failover 167
 - connection retry 169
 - copying data sources 142
 - creating data sources 133
 - data types supported
 - DB2 UDB on Linux/UNIX/Windows 457
 - Informix 463
 - Oracle 475
 - Sybase 481
 - deleting data sources 141
 - enabling SSL encryption on the ADO
 - Client 136
 - modifying data sources 140
 - renaming data sources 141
- AllowPartiallyTrustedCallers 406
- alternate servers, specifying
 - .NET Client 370
 - ADO Client 157
 - JDBC Client 238
 - ODBC Client 39
- API functions 85
- application
 - logging DataDirect Spy calls 551
 - optimizing performance for ODBC 105
 - performance tips for .NET 412
 - performance tuning for JDBC 344
- application developers
 - .NET 389
 - ADO 171
 - JDBC 327
 - ODBC 84
- application ID
 - generating automatically 100, 209
 - specifying explicitly 99, 158, 208
 - specifying for ADO data provider 158
 - specifying for ADO provider 208
 - specifying for JDBC driver 336
- ApplicationID ODBC connection attribute 62
- ApplicationName
 - ADO Client 158
 - JDBC Client 237
 - ODBC Client 62
- arguments, null 106
- Array object 489
- arrays of parameter
 - support
 - .NET data provider 379
- arrays of parameters
 - performance advantages 116
 - support
 - ODBC driver 101
- ASCII
 - converting to character 439
 - function 439
- ASIN function 442
- assembly name of the .NET provider 378
- Asynchable
 - Abort 194
 - Commit 194

- ATAN function 442
- ATAN2 function 442
- attributes
 - ADO 156
 - DistinguishedName 164
 - ODBC 61
- Autocommit Isolation Levels 194
- AutomaticApplicationID ODBC connection
 - attribute 62

B

- balancing the connection load 377
- BigDecimal objects, fetching 346, 491
- binding SQL statements
 - JDBC Client 329
 - ODBC Client 88
- blanks, generating 441
- Blob object 490
- BlockFetchForUpdate 62
- Blocking Storage Objects 190, 203
- Bookmark Information 203
- Bookmark Type 203
- bookmarks 184
- bound columns 113
- bridge, performance impact of using 422

C

- CallableStatement object 490
- calling a data source in a JDBC
 - application 227
- cancelling functions (JDBC driver) 332
- catalog functions
 - defined 105
 - effect on performance 106
- Catalog Location 194
- Catalog Term 194
- Catalog Usage 195
- CEILING function 442

- centralized system information file 56
- Change Inserted Rows 190, 203
- changing directories for ADO client data
 - sources 143
- CHAR function 439
- checking version of
 - DataDirect Spy 325
- cipher suites 240
- Client
 - SequeLink for .NET 363
 - SequeLink for ADO 127
 - SequeLink for JDBC 215
 - SequeLink for ODBC 31
- client code page
 - See code pages
- client data sources
 - configuring
 - ODBC file 42
 - ODBC User and System 33
 - on Linux/UNIX 53
 - creating
 - ADO 133
 - ODBC 32
 - deleting ADO 141
 - modifying ADO 141
- client errors, SequeLink (.NET) 407
- client load balancing
 - .NET Client 373
 - about 377
 - ADO data provider 168
 - JDBC driver 257
 - ODBC driver 81
 - see also *SequeLink Administrator's Guide*
- Clob object 501
- closing opened connections (.NET) 400
- code examples (.NET)
 - calling a stored procedure 591
 - limiting the rows returned by a Select
 - statement 588
 - retrieving a result set 587
 - retrieving a scalar value 594
 - retrieving warning information 593
 - updating data in a DataSet 589
 - using a distributed transaction 583

- using a local transaction with a
 - DataReader 582
 - using the CommandBuilder 586
- code page support
 - AIX 573
 - checking values for
 - IANAAppCodePage 566
 - HP 572
 - IANAAppCodePage attribute 569
 - Linux 574
 - Solaris 571
 - Windows 576
- code page support for ODBC Client 37
- Column Definition 195
- Column Privileges 191, 203
- COM Object Support 195
- Command object (.NET)
 - calling stored procedures 591
 - properties 397
 - threading support 395
- Command object (ADO)
 - dynamic properties 190
 - methods 190
- CommandBuilder class
 - impact on performance 415
 - performance implications 415
- CommandBuilder object
 - code example 586
 - properties 398
- commands
 - retrieving little or no data 419
 - using multiple times 421
- committing data 121
- compatibility of JDBC versions 489
- compiler requirements, Linux/UNIX 84
- CONCAT function 440
- concurrency types for result sets 334
- Configuration Manager
 - menu bar 130
 - overview 128
- configuring
 - ADO client data sources 133
 - file client data sources 42
 - JDBC data sources 225
 - ODBC client data source for Linux/UNIX 53
- connecting
 - example using Driver Manager 318
 - improving performance 416
 - overview 364
 - resetting the state of the connection 371
 - to data source, logon dialog box 147
 - tracking performance 410
 - using DataDirect Test 267
 - using URLs (JDBC) 224
 - with a provider string 155
 - with JDBC data sources 236
- connection
 - attributes in ADO 156
 - attributes in ODBC 61
 - defining using master data source file (ADO) 145
 - handles 121
 - JDBC options 215
 - managing 120
 - non-pooled 549
 - ODBC options 31
 - options in .NET 369
 - precedence for JDBC driver 236
 - properties for JDBC driver 235
 - testing ADO 147
 - testing ODBC on Windows 52
- Connection dialog box 147
- connection failover
 - .NET data provider
 - Connection Retry Count connection option 371
 - Connection Retry Delay connection option 372
 - Load Balancing connection option 373
 - overview 375
- ADO Client 164

- ADO data provider
 - Alternate Servers attribute 157
 - Connection Retry Count attribute 159
 - Connection Retry Delay attribute 160
 - defining on the Options tab 139
 - LDAP server 161
- client load balancing and 377
- JDBC driver
 - ConnectionRetryCount connection property 241
 - ConnectionRetryDelay connection property 242
 - LoadBalancing connection property 248
 - overview 254
- ODBC driver
 - AlternateServers connection attribute 79
 - ConnectionRetryCount connection properties 79
 - ConnectionRetryDelay connection attribute 79
 - LDAP server 80
 - LoadBalancing connection attribute 80
 - overview 76
- Connection Lifetime 370
- Connection object
 - .NET
 - properties 399
 - threading 395
 - using in the namespace 389
 - ADO
 - dynamic properties 194
 - methods supported 193
 - JDBC 501
- connection pool
 - adding connections (.NET) 365
 - closing (JDBC) 550
 - creating (.NET) 364
 - handling dead connections (.NET) 367
 - handling distribution transactions (.NET) 368
 - improving performance (JDBC) 356
 - tracking performance (.NET) 410
- Connection Pool Manager, about (JDBC) 218
- connection pooling
 - .NET data provider
 - adding connections 365
 - creating 364
 - enabling 374
 - maximum number of pools 373
 - minimum number of connections 374
 - removing connections 366
 - JDBC driver
 - connecting to a data source 547
 - Connection Pool Manager 541
 - creating a data source 544
 - creating a pooled data source object 542
 - terminating the pool manager 550
 - using 229
 - ODBC driver 121
- connection retry
 - ADO data provider 169
 - JDBC driver 258
 - ODBC driver 82
- Connection Retry Count, specifying
 - .NET data provider 375
 - ADO data provider 159
 - JDBC driver 256
 - ODBC driver 78
- Connection Retry Delay, specifying
 - .NET data provider 376
 - ADO data provider 160
 - JDBC driver 256
 - ODBC driver 78
- Connection Status 195
- contacting Technical Support 27
- conventions used 25
- converting variant types to date/time types 208
- copying, ADO client data sources 142
- COS function 442
- COT function 442
- counters, PerfMon 410
- creating
 - ADO client data sources 133
 - connection string for .NET provider 369
 - JDBC data sources 226

- master data source file 145
- ODBC Client data sources 32
- template data source file 144
- CURDATE function 444
- Current Catalog 195
- cursors
 - forward-only 401
 - forward-only result sets (JDBC) 333
 - keyset-driven scrollable (ODBC) 93
 - scroll-insensitive (Java 2 platform) 260
 - scroll-insensitive result sets (JDBC) 333
 - scroll-sensitive result sets (JDBC) 333
 - static scrollable (ODBC) 93
 - support for scrollable (JDBC) 334
 - support for scrollable (ODBC) 93
 - using scrollable instead of cursor library (ODBC) 119
- CURTIME function 444

D

- data shaping 206
- data source
 - calling in a JDBC application 227
 - configuring
 - JDBC 225
 - on Linux/UNIX (ODBC) 53
 - User and System (ODBC) 33
 - values overridden 207
 - connecting with JDBC 236
 - creating
 - ADO client 133
 - JDBC 226
 - non-pooled 542
 - ODBC file client 42
 - ODBC system 33
 - ODBC user 33
 - pooled 542
 - default 143
 - displaying properties 131
 - examples of JDBC 227
 - file
 - creating a master 145
 - creating a template 144
 - non-pooled 542
 - system 33
 - user 33
 - data source information property group 176
 - Data Source Name 195
 - Data Source Object Threading Model 195
 - Data Source property 195
 - data source property group 176
 - data types
 - .NET 390
 - choosing to improve performance 425
 - DB2 UDB on Linux/UNIX/Windows 456
 - DB2 UDB on z/OS 452
 - Informix 462
 - mapping year format 208
 - Microsoft SQL Server 466
 - Oracle 473, 478
 - Sybase 480
 - DataAdapter object
 - code example 587
 - properties 400
 - database
 - data dictionary 109
 - meta-information, retrieving 109
 - naming using JNDI 228
 - Database connection attribute
 - .NET Client 372
 - ADO provider 160
 - ODBC driver 64
 - Database Data Dictionary
 - filters 110
 - views (DB2 UDB for z/OS) 110
 - database errors (.NET client) 408
 - DATABASE function 447
 - DatabaseMetaData object 505
 - database-specific information 451
 - DataDirect Configuration Manager
 - menu bar 130
 - DataDirect Connection Pool Manager 221

- DataDirect Spy
 - about 315
 - attributes 323
 - checking the version 325
 - enabling
 - using DataDirect Spy URL 320
 - using JDBC data sources 318
 - using JDBC Driver Manager 317
 - example
 - JDBC data source connection 320
 - JDBC Driver Manager
 - connection 317, 318
 - URL 320
 - generating a log 551
 - jar file 222
 - log example 553
 - logging
 - generating a log 551
 - turning on and off 551
 - options 323
 - overview 217
 - registering the JDBC driver 322
 - setEnabledLogging method 551
 - SpyAttributes connection property 317
 - URL syntax 323
 - using with JDBC data sources 324
- DataDirect Test
 - batch execution on a prepared statement 284
 - configuring 264
 - connecting 267
 - establishing savepoints 290
 - executing a prepared statement 274
 - executing a simple select statement 272
 - files 221
 - LOB support 309
 - retrieving database metadata 278
 - returning ParameterMetaData 288
 - scrolling through a result set 281
 - starting 265
 - tutorial 263
 - updateable result sets 297
 - using 263
- DataReader object
 - choosing when to use 412
 - code example 580, 582
 - importance of closing 401
 - properties 401
- DataSet
 - choosing when to use 412
 - code example 589
 - creating and using 427
 - keeping result sets small 412
- DataSource object 516
- date
 - returning current 444
 - returning month 445
 - returning year 446
 - scalar functions 444
- DAYNAME function 444
- DAYOFMONTH function 445
- DAYOFWEEK function 445
- DAYOFYEAR function 445
- DB2 UDB for z/OS
 - data types supported
 - ODBC driver 452
- DB2 UDB on Linux/UNIX/Windows
 - data types for the .NET provider 460
 - data types for the JDBC driver 459
 - Distributed Transaction Management
 - support 231
 - scalar functions 434
 - support for scrollable cursors
 - JDBC 334
 - ODBC 93
- DB2 UDB on Windows and UNIX
 - Database Data Dictionary views 110
- DB2 UDB on z/OS
 - Database Data Dictionary views 110
 - scalar functions 433
- DBMS Name property 195
- DBMS Version property 195
- DBPassword 64
- DBPROP_
 - ROWSETCONVERSIONSONCOMMAND 180
- DBPROP_ SUBQUERIES 181
- DBPROP_ SUPPORTEDTXNDL 181

DBPROP_SUPPORTEDTXNISOLEVELS 181	DBPROP_INIT_LCID 182
DBPROP_SUPPORTEDTXNISORETAIN 181	DBPROP_INIT_MODE 182
DBPROP_ABORTPRESERVE 184	DBPROP_INIT_OLEDBSERVICES 183
DBPROP_ACCESSORDER 184	DBPROP_INIT_PROMPT 182
DBPROP_ASYNCCTXNABORT 177	DBPROP_INIT_PROVIDERSTRING 182
DBPROP_ASYNCCTXNCOMMIT 177	DBPROP_LITERALBOOKMARKS 186
DBPROP_AUTH_PASSWORD 182	DBPROP_LITERALIDENTITY 186
DBPROP_AUTH_USERID 182	DBPROP_LOCKMODE 186
DBPROP_AUTH_PASSWORD 182	DBPROP_MAXINDEXSIZE 178
DBPROP_AUTH_PERSIST_SENSITIVE_AUTHIN FO 182	DBPROP_MAXOPENROWS 186
DBPROP_AUTH_USERID 182	DBPROP_MAXPENDINGROWS 186
DBPROP_BLOCKINGSTORAGEOBJECTS 184	DBPROP_MAXROWS 186
DBPROP_BOOKMARKINFO 184	DBPROP_MAXROWSIZE 178
DBPROP_BOOKMARKS 184	DBPROP_MAXROWSIZEINCLUDESBLOB 179
DBPROP_BOOKMARKSKIPPED 184	DBPROP_MAXTABLEINSELECT 179
DBPROP_BOOKMARKTYPE 184	DBPROP_MEMORYUSAGE 186
DBPROP_CANFETCHBACKWARDS 184	DBPROP_MULTIPLEPARAMSETS 179
DBPROP_CANSROLLBACKWARDS 184	DBPROP_MULTIPLERESULTS 179
DBPROP_CATALOGLOCATION 177	DBPROP_MULTIPLESTORAGEOBJECTS 179
DBPROP_CATALOGTERM 177	DBPROP_MULTITABLEUPDATE 179
DBPROP_CATALOGUSAGE 177	DBPROP_NULLCOLLATION 179
DBPROP_COLUMNDEFINITION 177	DBPROP_OLEOBJECTS 179
DBPROP_COMMITPRESERVE 185	DBPROP_OPENROWSETSSUPPORT 179
DBPROP_CONCATNULLBEHAVIOR 178	DBPROP_ORDERBYCOLUMNSINSELECT 179
DBPROP_CONNECTIONSTATUS 178	DBPROP_OTHERINSERT 186
DBPROP_CURRENTCATALOG 176	DBPROP_OTHERUPDELETEDELETE 186
DBPROP_DATASOURCENAME 178	DBPROP_OUTPUTPARAMETERAVAILABILITY 179
DBPROP_DATASOURCEREADONLY 178	DBPROP_OWNINSERT 186
DBPROP_DBMSNAME 178	DBPROP_OWNUPDELETEDELETE 186
DBPROP_DBMSVER 178	DBPROP_PERSISTENTIDTYPE 179
DBPROP_DELAYSTORAGEOBJECTS 185	DBPROP_PREPAREABORTBEHAVIOR 180
DBPROP_DSOTHRADMODEL 178	DBPROP_PREPARECOMMITBEHAVIOR 180
DBPROP_HETEROGENEOUSTABLES 178	DBPROP_PROCEDURETERM 180
DBPROP_IDENTIFIERCASE 178	DBPROP_PROVIDERFRIENDLYNAME 180
DBPROP_IMMOBILEROWS 185	DBPROP_PROVIDERNAME 180
DBPROP_INIT_DATASOURCE 182	DBPROP_PROVIDEROLEDBVER 180
DBPROP_INIT_HWND 182	DBPROP_PROVIDERVER 180
DBPROP_INIT_MODE 183	DBPROP_QUOTEDIDENTIFIERCASE 180
DBPROP_INIT_PROMPT 183	DBPROP_REENTRANTEVENTS 186
DBPROP_INIT_PROVIDERSTRING 183	DBPROP_REMOVEDELETED 187
DBPROP_INIT_CATALOG 182	DBPROP_REPORTMULTIPLECHANGES 187
DBPROP_INIT_DATASOURCE 182	DBPROP_RETURNPENDINGINSERTS 187
DBPROP_INIT_HWND 182	DBPROP_ROWRESTRICT 187

- DBPROP_ROWTHREADMODEL 187
- DBPROP_SCHEMATERM 180
- DBPROP_SCHEMAUSAGE 180
- DBPROP_SERVERCURSOR 187
- DBPROP_SERVERNAME 180
- DBPROP_SESS_ AUTOCOMMITISOLEVELS 188
- DBPROP_SQLSUPPORT 180
- DBPROP_STRONGIDENTITY 187
- DBPROP_STRUCTUREDSTORAGE 180
- DBPROP_TABLETERM 181
- DBPROP_TRANSACTEDOBJECT 187
- DBPROP_UNIQUEROWS 187
- DBPROP_UPDATABILITY 187
- DBPROP_USERNAME 181
- dead connections 400
- debugging
 - using PerfMon 410
 - using the Trace object to maintain security 408
- default data source files
 - master 143
 - template 143
- Default Length for Long Data 161
- defining
 - data source using the ODBC Administrator 33
 - default setup options 143
- DEGREES function 442
- Delay Storage Object Updates 203
- deletes, positional 122
- deleting ADO client data sources 141
- developing ODBC applications 84
- diagnostic support
 - PerfMon counters 410
 - tracing method calls 408
- DIFFERENCE function 440
- direct SSL encryption 244
- directory structure for the JDBC Client 220
- Distinguished Name identifier 161
- DistinguishedName attribute
 - ADO 164
 - ODBC 64
- Distributed Transaction Management 231

- distributed transactions
 - code example 583
 - for more information 427
 - in a .NET connection pool 368
- documentation, SequeLink 22
- double-byte character sets, using on UNIX and Linux 565
- downloading applets, tips 344
- DPROP_GROUPBY 178
- Driver object 516
- DSN connection attribute 65
- dynamic parameters, binding
 - JDBC driver 329
 - ODBC Client 88
- dynamic properties
 - Command object (ADO) 190
 - Connection object (ADO) 194
 - Recordset object (ADO) 203

E

- EnableDescribeParam 65
- enabling tracing 408
- encryption, SSL
 - ADO Client 136
 - JDBC Client 244
 - ODBC Client 65
- environment variables (Linux/UNIX) 55
- environmental variable use in tracing 409
- environment-specific information 26
- error handling
 - .NET data provider 406
 - ADO data provider 209
 - JDBC driver 342
 - ODBC driver 103
- Error object (.NET) 401
- ErrorCollection object
 - .NET data provider 401
 - checking for multiple errors 402
- event handling for the ADO.NET data provider 396

- examples
 - 64-bit ODBC Client configured for Solaris 54
 - connection pooling 229
 - creating and using JDBC data sources 227
 - DataDirect Spy
 - JDBC data source connection 320
 - JDBC Driver Manager
 - connection 317, 318
 - URL 320
 - DataDirect Spy log 553
 - Driver Manager, using to connect 317, 318
 - odbc.ini file configured for Solaris 54
- Exception object (.NET) 402
- ExecuteScalar and ExecuteNonQuery, performance implications 419
- EXP function 442
- ExtLogControl class 552
- EXTRACT function 445

F

- fetching
 - backward 184, 191, 203
 - BigDecimal objects 346
- fetching random data 412
- FetchNextOnly 66
- file
 - creating a master data source 145
 - creating a template data source 144
- file client data sources, configuring 42
- File System JNDI Provider 222
- files in the JDBC Client directory 220
- filters, Database Data Dictionary 110
- FixCharTrim 66
- FLOOR function 443
- forward only cursors 114
- forward-only result sets (JDBC) 333
- Framework types (.NET) 390
- FullTrust 405

- functions
 - cancelling in multithreaded applications 92, 332
 - date and time 444
 - numeric 442
 - ODBC API 85
 - scalar 432
 - string 439

G

- generating DataDirect Spy log 551
- generating ODBC application IDs
 - automatically 100
- get methods, using effectively 354
- GetOutputParams 67
- Global Assembly Cache (GAC) 378
- GROUP BY Support 196

H

- header files and libraries,
 - platform-specific 84
- Heterogeneous Table Support 196
- HLogonID 67
- Hold Rows 191, 203
- Host attribute 67, 161
- host password 67
- Host Password attribute 162
- HOURL function 445
- HPassword 67

I

- IANAAppCodePage, supported values 569
- Identifier Case Sensitivity 196
- IFNULL function 447
- Immobile Rows 204
- Informix
 - data types 462
 - persisting result set as XML 101
 - support for scrollable cursors (ODBC) 93
- Initial Catalog 196
- initialization properties 182
- INSERT function 440
- interfaces supported by ADO provider 172
- IOpenRowset 207
- IPersistFile 207
- IRowset interface 172
- IRowsetIdentity 183
- isolation levels
 - ADO data provider 197
 - ADO.NET data provider 395
 - by data store 486
 - JDBC driver 331
 - ODBC driver 91
- Isolation Retention 197
- ivslkcheckcp utility 566

J

- JAR files 344
- Java 2 Platform 259
- Java Transaction API 231
- Java Virtual Machine versions 489
- JavaDoubleToString 247
- JCA
 - resource adapter class 329
 - resource adapters
 - overview 232
 - using from an application 233
 - using with application server 233

JDBC

- compatibility 489
- connection options 215
- forward-only result sets 333
- functionality supported 489
- scroll-insensitive result sets 333
- scroll-sensitive result sets 333
- support 489
- support for 2.0 functionality 328
- JDBC 2.0 Optional Package 220, 222
- JDBC Client
 - 64-bit platform support 215
 - configuring JDBC data sources 226
 - connection properties
 - JavaDoubleToString 247
 - data types supported
 - DB2 UDB on z/OS 454
 - directory structure 220
 - overview 215
 - using connection pooling 229
 - using on a Java 2 Platform 259
 - using the JTA 231
- JDBC data sources 220
- JDBC driver
 - binding dynamic parameters 329
 - calling a data source in an application 227
 - client load balancing 257
 - configuring data sources 225
 - connection failover, configuring 254
 - connection properties 235, 237
 - connection retry 258
 - creating data sources 226
 - data types supported
 - DB2 UDB on Linux/UNIX/Windows 459
 - DB2 UDB on z/OS 452
 - Informix 462
 - Microsoft SQL Server 466
 - Oracle 473
 - Sybase 480
 - error handling 342
 - functionality for JDBC objects 489
 - loading 223
 - overview 216
 - performance 513

- registering with JDBC driver Manager 223
- ResultSet metadata 339
- specification supported 216
- specifying application IDs 336
- specifying connection URLs 224
- supported JDBC connection properties 237
- threading for driver 331
- tracking calls 217
- using
 - data sources 236
 - getObject method 354
- using connection pooling 229
- using Distributed Transaction Management 231
- using Spy with data sources 324
- JDBC Driver Manager
 - registering DataDirect Spy driver 320
 - SpyAttributes property, specifying with 316
- JNDI
 - registering a SequeLink for JDBC DataSource object 542
 - using for naming databases 228
- JNDI 1.2 222
- JTA 1.0.1 222

K

- keyset-driven scrollable cursors (ODBC) 93, 95

L

- LCASE function 440
- LDAP
 - configuring the ADO Client 135
 - configuring the ODBC Client 36, 53
 - creating a JDBC data source 227
 - property in ADO 164
 - setting parameters in DataDirect Test 264

- specifying port for the listener 71, 135, 163
- TCP/IP address of the LDAP server 67
- UseLDAP attribute (ADO) 164
- UseLDAP attribute (ODBC) 72
- LDAP JNDI Provider 222
- LEFT function 440
- LENGTH function 440
- libraries and header files, platform-specific 84
- license file, defining path for .NET data provider 373
- LicensePath 373
- limiting the rows returned by a Select statement 588
- Linux
 - See Linux/UNIX
- Literal Bookmarks 204
- literal parameter values 379
- Literal Row Identity 191, 204
- load balancing
 - See client load balancing
- Load Balancing connection attribute (ADO data provider) 162
- Load Balancing connection string option (.NET data provider) 373
- LoadBalancing connection attribute (ODBC driver) 70
- LoadBalancing connection property (JDBC driver) 248
- loading the JDBC driver 223
- local transactions 381
- Locale Identifier 197
- LOCATE function 440
- Lock Mode 191, 204
- log for DataDirect Spy, generating 551
- LOG function 443
- LOG10 function 443
- logging JDBC calls 315
- LogonID 70
- long data, performance issues with retrieving 111
- LTRIM function 440

M

- managed code
 - performance advantages 422
 - use in distributed transaction processing 382
 - use in local transactions 381
- ManagedConnectionFactory 329
- managing
 - connections to improve driver performance 120
 - JDBC data sources 226
 - retrieval of database meta-information 109
- mapping data types
 - ADO provider 208
 - SequeLink DbTypes 394
 - supported by DB2 UDB on Linux/UNIX/Windows 456
 - supported by DB2 UDB on z/OS ODBC driver 452
 - supported by Informix 462
 - supported by Microsoft SQL Server 466
 - supported by Oracle 473, 478
 - supported by Sybase 480
- master data source file
 - creating a 145
 - overview 143
- Maximum Index Size 197
- Maximum Open Chapters 197
- Maximum Open Rows 191, 204
- Maximum Pending Rows 191, 204
- Maximum Row Size 197
- Maximum Row Size Includes BLOB 197
- Maximum Rows 191, 204
- Maximum Tables in SELECT 197
- Memory Usage 191, 204
- metadata
 - parameter 337
 - ResultSet 339
- meta-information
 - limiting amount to be retrieved 110
 - managing retrieval of 109

- methods
 - .NET provider-specific 399
 - ADO Command object 190
 - ADO Recordset object 201
 - ExtLogControl class 552
 - supported by ADO Connection object 193
 - supported public 396
 - tracing calls 408
- Microsoft SQL Server, data types
 - supported 466
- MINUTE function 445
- MOD function 443
- Mode property 197
- modifying ADO client data sources 140
- MONTH function 445
- MONTHNAME function 445
- Multiple Connections property 197
- Multi-Table Update 197
- multithreaded applications
 - cancelling functions 92, 332
 - functionality of the ODBC driver 92

N

- namespace, .NET data provider 389
- naming databases with JNDI 228
- native managed providers, performance
 - advantages 422
- NewPassword 71
- non-pooled data source 542
- NOW function 445
- null arguments, impact on performance 106
- NULL Collation Order 198
- NULL Concatenation Behavior 198
- null values in data conversions 242
- Number
 - raising to power 443
 - rounding 443
- numeric functions 442

O

- object (.NET)
 - exposing 378
 - provider-specific prefix 396
 - SequeLinkCommand 397
 - SequeLinkCommandBuilder 398
 - SequeLinkConnection 399
 - SequeLinkError 401
 - SequeLinkErrorCollection 401
 - SequeLinkException 402
 - SequeLinkInfoMessageEventArgs 402
 - SequeLinkTrace 403
 - SequeLinkTransaction 381
- object (JDBC)
 - ExtConnection 552
 - fetching BigDecimal 346
- objects supported
 - by the .NET data provider 396
 - by the ADO data provider 172
 - by the JDBC driver 489
- Objects Transacted 192, 204
- ODBC
 - functions that improve driver performance 115
 - keyset-driven scrollable cursors 95
 - Level 2.x API functions 85
 - optimizing performance 105
 - static scrollable cursors 93, 94
- ODBC Administrator
 - configuring file client data sources 42
 - configuring User and System client data sources 33
 - starting 32
- ODBC driver
 - application IDs 99
 - binding dynamic parameters 88
 - cancelling functions in multithreaded applications 92, 332
 - client load balancing 81
 - configuring connection failover 76
 - configuring file client data sources 42
 - connecting to a data source using a logon dialog box 46
 - connection attributes 61
 - connection options 31
 - connection properties for connection failover 79
 - connection retry 82
 - enabling SSL encryption 65
 - error handling 103
 - handling Unicode characters 72
 - ODBC API functions supported 85
 - overview 31
 - persisting result set as XML data file 101
 - required libraries and header files 84
 - specification supported 31
 - SSL encryption 36
 - threading 91
 - using the system information file 53
- ODBC translators 37
- odbc.ini (Linux/UNIX)
 - sample odbc.ini file 54
 - using a centralized odbc.ini file 56
- odbc64.ini (Linux/UNIX)
 - configuration example 54
 - executing a shell script 55
- ODBCINI environment variable (Linux/UNIX) 55
- ODBCTest 98
- OLE DB
 - mapping methods to ADO 190
 - objects supported by the SequeLink for ADO provider 172
 - property groups supported 175
 - schema rowsets supported 174
- OLE DB Services property 198
- OLE DB Version property 198
- Open Rowset Support 198
- Oracle
 - data types 473
 - support for scrollable cursors (JDBC) 334
 - support for scrollable cursors (ODBC) 94
 - using stored procedures with (ODBC) 95
- ORDER BY Columns In Select List 198
- Others' Changes Visible 192, 204

Others' Inserts Visible 192
 Output Parameter Availability 198
 Own Changes Visible 192, 204
 Own Inserts Visible 192, 204

P

parameter (.NET)
 array support 379
 in stored procedures 380
 markers, using the ? symbol 379
 object, adding to a collection 379
 types 392
 parameter markers
 as arguments to stored procedures
 (.NET) 413
 ParameterCollection object 379
 parameter metadata (JDBC) 288
 INSERT and UPDATE statements 337
 SELECT statements 338
 ParameterMetaData object 330, 517
 Pass By Ref Accessors 198
 Password
 .NET data provider connection string
 option 374
 ADO data provider connection
 attribute 163
 ODBC driver connection attribute 71
 property for Connection object 198
 password, concealed in trace log 408
 PerfMon support 410
 performance hints
 .NET data provider
 size of data retrieved 424
 using a SequeLinkCommandBuilder
 object 398
 choosing the data type 425, 451
 committing data 121

JDBC
 fetching BigDecimal objects 346
 penalty for emulating multiple open
 ResultSet objects 513
 reducing download time 344
 using get methods effectively 354
 ODBC driver
 avoiding the cursor library 119
 catalog functions 105
 locking a row when isolation level is
 Read committed 62
 managing connections 120
 managing retrieval of database
 meta-information 109
 null arguments 106
 reducing the size of retrieved data 111
 retrieving data with
 SQLExtendedFetch 114
 using a dummy query 108
 resetting the state of a connection 371
 using bound columns 113
 performance optimization
 avoiding distributed transactions 419
 avoiding use of CommandBuilder
 objects 415
 choosing between a DataReader and a
 DataSet 412
 designing JDBC applications 355
 managing connections 416
 retrieving data 349
 selecting JDBC objects and methods 351
 turning off autocommit 418
 updating data 357
 using disconnected DataSet 412
 using native managed providers 422
 persisting
 information 207
 result set as XML data file (ODBC) 101
 PI function 443
 platform-specific header files and libraries 84
 pool manager
 See connection pooling
 pooled data source 542
 PooledConnection object 517

- Port (ODBC connection attribute) 71
- Port attribute 163
- positional updates and deletes 122
- POWER function 443
- precedence of JDBC connection
 - properties 236
- prefix on a public object 396
- Prepare Abort Behavior 198
- Prepare Commit Behavior 198
- prepared statements 421
- PreparedStatement object 518
- Preserve on Abort 192, 205
- Preserve on Commit 192, 205
- private assembly 378
- Procedure Term 198
- Prompt property for Connection object 198
- properties
 - ADO Command object 190
 - ADO Connection object 194
 - ADO Recordset object 203
 - Data Source Information property
 - group 176
 - Data Source property group 176
 - displaying ADO provider data source 131
 - Initialization property group 182
 - JDBC connection 235, 237
 - Oracle data provider 397
 - Session property group 188
 - supported public 396
- Property 183
- property groups supported 175
- Provider Friendly Name 198
- Provider Name 199
- provider string 155
- Provider Version 199
- Proxy Server, SequeLink 216
- pseudo-columns 123

Q

- QUARTER function 445
- QueryTimeout connection property 251
- Quick Restart 192, 205
- Quoted Identifier Sensitivity 199

R

- RADIANS function 443
- RAND function 443
- random data, fetching 412
- Read Only Data Source 199
- read-only result set 334
- Recordset object
 - dynamic properties 203
 - methods 201
 - overview 201
- Ref object 523
- Referenceable object 524
- registering the JDBC driver 223
- Remove Deleted Rows 192, 205
- removing connections from a connection
 - pool 366
- renaming an ADO Client data source 141
- REPEAT function 441
- REPLACE function 441
- Report Multiple Changes 192, 205
- resetting connection state 371
- resource adapters, JCA
 - overview 232
 - resource adapter class 329
 - support 219
- Resource Archive 220
- result sets
 - concurrency types (JDBC) 334
 - impact of size on scalability (.NET) 413
 - types (JDBC) 333
 - updatable supported (JDBC) 335
- ResultSet metadata for Select statements
 - (JDBC) 339

- ResultSet object 524
- ResultSetMetaData object 533
- resultSetMetaDataOptions performance considerations 252
- retrieving
 - data
 - avoiding long data 111, 423
 - reducing the size 111
 - using bound columns 113
 - using SQLExtendedFetch 114
 - database meta-information 109
 - long data using JDBC applications 349
 - warning information from .NET application 593
- Return Pending Inserts 192, 205
- reusing connections with connection pooling 229
- RIGHT function 441
- ROUND function 443
- Row Privileges 192, 205
- Row Threading Model 192, 205
- rowset
 - characteristics 183
 - properties 183
 - using 207
- Rowset Conversions on Command 199
- RowSet object 534
- rowset support 341
- RTRIM function 441

S

- sample code
 - calling a stored procedure 591
 - limiting the rows returned by a Select statement 588
 - retrieving a result set using a DataAdapter object 587
 - retrieving a scalar value 594
 - retrieving warning information 593
 - updating data in a DataSet 589
 - using a distributed transaction 583
 - using a local transaction with a DataReader 582
 - using the CommandBuilder 586
- SavePoint object (JDBC) 535
- savepoint support in .NET data provider 405
- saving current connection information 207
- scalar functions
 - .NET code example of retrieving scalar values 594
 - support for 432
- schema rowsets supported 174
- Schema Term 199
- Schema Usage 199
- Scroll Backward 192, 205
- scrollable cursors
 - concurrency types (JDBC) 334
 - insensitive (JDBC) 260
 - result set types (JDBC) 333
 - using keyset-driven (ODBC) 95
 - using static (ODBC) 94
 - using with JDBC driver 333
 - using with ODBC driver 93
- scrolling backward 184
- scroll-insensitive result sets (JDBC) 333
- scroll-sensitive result sets (JDBC) 333
- SECOND function 445
- security (.NET)
 - attributes 406
 - enabling for tracing 408
 - required permissions for data provider 405
 - see also *SequeLink Administrator's Guide*
 - SequeLinkTrace object 403
- SequeLink Client for .NET
 - data types
 - DB2 UDB on Linux/UNIX/Windows 460
 - DB2 UDB on z/OS 455
 - Informix 465
 - Microsoft SQL Server 471
 - Oracle 478
 - event handling 396
 - Framework data types 390
 - isolation levels 395
 - limiting number of rows returned 398
 - mapping parameter types 392

- namespace 382, 389
- public objects supported 396
- SequeLinkCommand object 397
- SequeLinkConnection object 389
- thread support 395
- tracing method calls 408
- using a distributed transaction 583
- SequeLink Client for ADO
 - ADO Command object 190
 - connecting to a data source 148
 - connection attributes 156
 - data shaping 206
 - data source information properties 176
 - data source property group 176
 - data types
 - DB2 on z/OS 452
 - DB2 UDB on Linux/UNIX/Windows 456
 - Informix 462
 - Microsoft SQL Server 466
 - Oracle 473, 478
 - Sybase 480
 - error handling 209
 - initialization properties 182
 - objects and interfaces 172
 - overview 127
 - rowset properties 183
 - rowset property group 183
 - session property group 188
 - SQL grammar supported 200
- SequeLink Client for JDBC
 - JDBC 3.0 functionality 328
 - Proxy Server 216
 - specifying application IDs 336
- SequeLink Client for ODBC
 - checking code page used by applications 566
 - configuring file client data sources 42
 - connection dialogs 46
 - data types
 - DB2 UDB on Linux/UNIX/Windows 456
 - DB2 UDB on z/OS 452
 - Informix 462
 - Microsoft SQL Server 466
 - Oracle 473
 - Sybase 480
 - functions that improve performance 115
 - isolation levels 486
 - overview 31
- SequeLink documentation 22
- SequeLink Proxy Server
 - files 221
 - overview 216
 - using SSL encryption 216
- SequeLink Server errors (.NET) 407
- SequeLink Server for DB2 UDB on z/OS
 - data types 452
 - scalar functions 433
- SequeLink Server for JDBC Socket
 - JTA support 231
 - keyset-driven cursor support 93
 - scroll-sensitive cursor support 334
- SequeLink Server for ODBC Socket
 - JTA support 231
 - scrollable cursor support (JDBC) 334
 - support for scrollable cursors (ODBC) 94
- SequeLinkCommand object 397
- SequeLinkCommandBuilder object 398
- SequeLinkConnection object 399
- SequeLinkError object 401
- SequeLinkErrorCollection 401
- SequeLinkException object 402
- SequeLinkInfoMessageEventArgs object 402
- SequeLinkTrace object 409
- SequeLinkTransaction object 381, 405
- Serializable object 535
- Server Cursor 192, 205
- server errors, SequeLink (.NET) 407
- Server Name 200
- ServerDatasource 375
- served components 427
- session property group 188

- sessions
 - maximum number supported 177
 - setting maximum number supported 194
- setEnableLogging, using to turn on and off DataDirect Spy logging 551
- setting
 - maximum number of sessions 194
 - ODBCINI environment variable (Linux/UNIX) 55
- shared assembly 378
- shell script, using to set environmental variables 55
- SIGN function 443
- SIN function 443
- Skip Deleted Bookmarks 205
- SLKStaticCursorLongColBuffLen 72
- Snapshot Isolation level, SQL Server Wire Protocol 488
- SOUNDEX function 441
- SPACE function 441
- specification supported
 - JDBC 216, 328
 - ODBC driver 31
 - OLE DB 180
- specifying alternate servers
 - ODBC Client 39
- specifying connection attributes 369
- specifying load balancing
 - ODBC Client 38
- specifying provider-specific logon information 196
- Spy
 - see DataDirect Spy
- SQL support
 - binding SQL statements (JDBC) 329
 - date and timestamp escape sequences 432
 - numeric functions 442
 - outer join escape sequences 448
 - overview 431
 - scalar functions 432
 - string functions 439
- SQL Support (ADO connection property) 200
- SQLCancel, effect of threading 92, 332
- SQLColumns, performance implications 108
- SQLExecDirect 115
- SQLExtendedFetch 114
- SQLParamOptions 118
- SQLPrepare 115
- SQLSetConnectAttr 100
- SQLSpecialColumns 123
- SQRT function 443
- SSL encryption
 - ADO Client 136
 - cancel functionality 332
 - connection URL format 224
 - JAR files 220
 - JDBC Client 244
 - ODBC Client 36
 - permissions required 260
 - using with the proxy server 216
- starting
 - Configuration Manager 131
 - ODBC Administrator 32
- statement handles 121
- Statement object 535
- static methods, enabling tracing with 409
- static scrollable cursors (ODBC) 93, 94
- Status bar (Configuration Manager) 131
- stored procedures
 - .NET code example 591
 - output array support 379
 - performance implications 421
 - support for (.NET) 380
 - using parameter markers as arguments 413
 - using the "?" in SQL statements (.NET) 379
 - using with ODBC driver 116
 - using with Oracle 95
- string
 - changing case of 440
 - functions 439
 - length of 440
 - removing blanks from 440, 441
 - returning substring of 441
- strong name 378
- Strong Row Identity 192, 205
- Struct object 538
- Structured Storage 200

- Subquery Support 200
- SUBSTRING function 441
- SupportLink 27
- Sybase
 - .NET code example 580, 587, 591
 - data types 480
 - support for scrollable cursors (ODBC) 94
- syntax for URLs for Spy 323
- system information file
 - centralized 56
 - using to configure the ODBC driver 53

T

- table characteristics, determining 108
- Table Term 200
- TAN function 443
- TCP port 224
- TCP/IP port for SequeLink listener 71
- Technical Support, contacting 27
- template data source file
 - creating a 144
 - overview 143
- terminating the pool manager 550
- testing
 - ADO connections 147
 - JDBC connections 267
 - ODBC connections on Windows 52
- threading
 - .NET provider 395
 - ADO provider 178, 195
 - JDBC driver 331
 - ODBC driver use 91
- time functions 444
- TIMESTAMPADD function 446
- TIMESTAMPDIFF function 446
- Trace object (.NET)
 - properties 403
 - using 408

- tracing
 - .NET data provider method calls 403
 - enabling
 - using environment variables 409
 - using static methods 409
 - overview 408
 - re-creating the trace file 408
 - setting the path to the trace file 408
- tracking JDBC calls 217, 324
- Transaction DDL 201
- transactions
 - isolation levels 197
 - managing commits 418
 - performance considerations of using
 - distributed 419
 - support for local (.NET) 381
- Translate button 37
- transliteration
 - ODBC Client 37
 - see also *SequeLink Administrator's Guide*
- troubleshooting driver problems 551
- TRUNCATE function 443
- tuning and debugging applications with
 - PerfMon 410
- turning on and off DataDirect Spy
 - logging 551
- typographical conventions 25

U

- UCASE function 441
- Unicode
 - ANSI code page for conversions 564
 - data types 563
 - DB2 UDB data types supported 456
 - function calls 559
 - JDBC driver 341
 - non-Unicode application 561
 - see also *SequeLink Administrator's Guide*
 - support for ADO data provider 457
 - support in ODBC drivers 559
 - Unicode application 560

- Unique Rows 193, 205
- UNIX and Linux
 - code pages, IANAAppCodePage attribute 569
 - compiler requirements 84
 - configuring an ODBC data source 53
 - Driver Manager 567
 - multithreading functionality of the ODBC driver 92
 - setting environmental variables 55
 - system information file 53
 - using a centralized odbc.ini or odbc64.ini file 56
 - using double-byte character sets 565
- unmanaged code
 - performance impact 422
 - use in distributed transaction processing 381
- Updatability 193, 205
- updatable result sets 334
- updates, positional 122
- updating opened rowset 183
- URL
 - format for connections (JDBC) 224
 - precedence of JDBC connection properties 236
- Use Bookmarks 193, 205
- Use LDAP attribute (ADO) 164
- UseLDAP attribute (ODBC) 72
- USER function 447
- User Name property of Connection object 201
- using
 - arrays of parameters 116
 - centralized odbc.ini files (Linux/UNIX) 56
 - Command.Prepare 421
 - JDBC Client on Java 2 Platform 259
 - keyset-driven cursors 93
 - scrollable cursors
 - with the JDBC driver 335
 - with the ODBC driver 93
 - SequeLink Client for ADO 127
 - static cursors 93
 - stored procedures with Oracle 95

- UTF-16, using for applications on Linux/UNIX 566

W

- warning information, retrieving 593
- WEEK function 446
- window handle 182
- Window Handle (ADO Connection object property) 201
- Windows
 - code page support 576
 - configuring SequeLink Client for ADO 127
 - multithreading functional of the ODBC driver 92
 - required ODBC libraries and header files 84
 - testing ODBC connections 52
- WorkArounds attribute 73

X

- XAConnection object 538
- XADatasource object 538
- XML
 - manipulating relational data as 413
 - persistence, implementing (ODBC driver) 101

Y

- year format for conversions 208
- YEAR function 446

Z

z/OS

- data types 452

 - .NET Client 455

 - ADO Client 453

 - JDBC Client 454

- data types supported

 - ODBC driver 452