DataDirect TECHNOLOGIES

SequeLink®

Developer's Reference

June 2003

© 2003 DataDirect Technologies. All rights reserved. Printed in the U.S.A.

DataDirect, DataDirect Connect, and SequeLink are registered trademarks, and Client/Server MiddleWare, DataDirect Connect Integrator, DataDirect jXTransformer, DataDirect Reflector, DataDirect SequeLink Integrator, DataDirect Spy, DataDirect Test, and SupportLink are trademarks of DataDirect Technologies.

Java, Java Naming and Directory Interface, and JDBC are registered trademarks of Sun Microsystems, Inc. in the United States and other countries. DataDirect Technologies is independent of Sun Microsystems, Inc.

All other trademarks are the property of their respective owners.

DataDirect products for UNIX platforms include:

ICU Copyright (c) 1995-2001 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.

DataDirect Connect *for* SQL/XML includes:

Xerces, developed by the Apache Software Foundation (http://www.apache.org). Copyright (C) 1999-2003 The Apache Software Foundation. All rights reserved.

Xalan, developed by the Apache Software Foundation (http://www.apache.org). Copyright (C) 1999-2003 The Apache Software Foundation. All rights reserved.

JDOM, developed by the JDOM Project (http://www.jdom.org). Copyright (C) 2001 Brett McLaughlin & Jason Hunter. All rights reserved.

DataDirect SequeLink includes:

Portions created by Eric Young are Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com). All Rights Reserved.

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, but only to the extent necessary to support the users authorized access to the software under the license agreement. Any reproduction of the documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication, or disclosure by the Government of the United States is subject to the restrictions as set forth in FAR52.227- 14 and DFAR252.227-7013 et seq. or the successor as appropriate. Manufacturer is DataDirect Technologies, 3202 Tower Oaks Blvd. Suite 300, Rockville, Maryland 20852. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

DataDirect Technologies
3202 Tower Oaks Blvd. Suite 300
Rockville, Maryland 20852

Table of Contents

List of Tables **13**

Preface **17**

What Is DataDirect SequeLink? 17

Using This Book 17

Other SequeLink Documentation 20

Conventions Used in This Book 22
Typographical Conventions 22
Environment-Specific Information 23

Contacting Technical Support 24

Part 1: Developing ODBC Applications

1 Using the ODBC Client **29**

About the ODBC Client 29

Using the ODBC Administrator 30

Configuring ODBC Client Data Sources on Windows 31
Configuring ODBC User and System Client Data Sources 31
Configuring ODBC File Client Data Sources 34
ODBC Connection Dialogs 39
Testing ODBC Connections on Windows 45

Configuring ODBC Client Data Sources on UNIX 46
Configuring the System Information File 46
Example: odbc.ini for Solaris 46
Example: odbc64.ini for Solaris 47

Setting Environment Variables 47
Using a Centralized System Information File 48

Connecting Using a Connection String 49

ODBC Connection Attributes 50

2 Developing ODBC Applications 63

Required ODBC Libraries and Header Files 64

UNIX Compiler Requirements 65

ODBC API Functions 65
Support for Unicode ODBC W (Wide) Function Calls 68

SQL Escape Sequences 68

Data Types and Isolation Levels 69

Threading 69
Threading Architecture 69
Cancelling Functions in Multithreaded Applications 71

Using Scrollable Cursors 72
Static and Keyset-Driven Cursors 72
Using Static Scrollable Cursors 73
Using Keyset-Driven Scrollable Cursors 73

Using Stored Procedures with Oracle 74

Specifying Application IDs 78
Specifying Application IDs Explicitly 78
Generating Application IDs Automatically 79

Persisting a Result Set as an XML Data File 80

Error Handling 82
SequeLink *for* ODBC Driver Errors 82
SequeLink Client Errors 82
SequeLink Server Errors 83
Database Errors 83

Developing Performance-Optimized ODBC Applications 84
Catalog Functions 84
Retrieving Data 90
ODBC Function Selection. 94
Designing ODBC Applications. 97
Updating Data 99

Part 2: Developing ADO Applications

3 Using the ADO Client 105

About the ADO Client 105

Using the DataDirect Configuration Manager 106
Working with the DataDirect Configuration Manager. . . 107
Displaying Data Source Properties 109

Configuring ADO Client Data Sources 111
Creating an ADO Client Data Source 112
Modifying an ADO Client Data Source. 115
Renaming an ADO Client Data Source 115
Deleting an ADO Client Data Source 116
Copying an ADO Client Data Source 116
Changing Data Source Directories 117
Defining Default Setup Options 118

Connecting to an ADO Client 121
Testing ADO Connections 121
ADO Connection Dialogs. 122
Connecting with a Provider String 128
ADO Connection Attributes 130

4 Developing ADO Applications 135

OLE DB Objects and Interfaces 136

Supported Schema Rowsets 138

Supported OLE DB Property Groups 139
Data Source Property Group 140
Data Source Information Property Group. 140
Initialization Property Group. 145
Rowset Property Group 147
Session Property Group 152

OLE DB Interfaces Supported in ADO 152

Mapping ADO Methods and Properties 154
ADO Command Object. 154
Connection Object 157
Recordset Object 165

Data Shaping 170

Persisting Information 171

Using Rowsets. 171

Unicode Support. 172

Mapping Data Types 172

Specifying Application IDs 173
Specifying Application IDs Explicitly 173
Generating Application IDs Automatically 173

Error Handling 174
SequeLink *for* ADO Provider Errors 174
SequeLink Client Errors 174
SequeLink Server Errors 175
Database Errors 175

Part 3: Developing JDBC Applications

5 Using the JDBC Client 179

About the JDBC Client 179
JDBC Driver 180
SequeLink Proxy Server 180
DataDirect Spy 181

DataDirect Test 182
DataDirect Connection Pool Manager 182
J2EE Connector Architecture (JCA) Resource Adapter . . . 183
JDBC Client Directory Structure 184

Loading the JDBC Driver 187

Specifying JDBC Driver Connection URLs 188

Configuring JDBC Data Sources 190
Creating and Managing JDBC Data Sources 191
Using JNDI for Naming Databases 191
Using Connection Pooling 192

Using the Java Transaction API 195

J2EE Connector Architecture (JCA) Resource Adapter 196
Using the Resource Adapter with an Application Server 197
Using the Resource Adapter from an Application 198

Specifying Connection Properties 200
Using Connection URLs or the JDBC Driver Manager 200
Using JDBC Data Sources 200

JDBC Connection Properties 201

Testing JDBC Connections 208

Using the JDBC Client on a Java 2 Platform 208

6 Using DataDirect Test 211

DataDirect Test Tutorial 211
Configuring DataDirect Test 212
Starting DataDirect Test 213
Connecting Using DataDirect Test 215
Executing a Simple Select Statement 221
Executing a Prepared Statement 223
Retrieving Database Metadata 227
Scrolling Through a Result Set 229
Batch Execution on a Prepared Statement 233

Returning ParameterMetaData 237
Establishing Savepoints 238
Updatable Result Sets 245
LOB Support 257

7 Tracking JDBC Calls 263

About Spy 263

Loading the Spy JDBC Driver 264

Spy URL Syntax and Spy Attributes 265

Using Spy with JDBC Data Sources 266

Spy URL Example 267

Spy Log Example. 267

8 Developing JDBC Applications 271

JDBC 3.0 Support 272

JCA Resource Adapter Class. 273

SQL Escape Sequences 273

Data Types and Isolation Levels. 273

Threading 274
Threading Architecture 274
Cancelling Functions in Multithreaded Applications 274

Using Scrollable Cursors. 276
Result Set Types 276
Concurrency Types 277
Using Scrollable Cursors. 278

Specifying Application IDs 279

Unicode Support. 280

Error Handling 281
SequeLink *for* JDBC Driver Errors 281
SequeLink Server Errors 281
Database Errors 282

Fine-Tuning JDBC Application Performance 283
Reducing Download Time 283
Fetching BigDecimal Objects 284
Using Database Metadata Methods 285
Retrieving Data 288
Selecting JDBC Objects and Methods 290
Designing JDBC Applications 293
Updating Data 295

Part 4: Developing .NET Applications

9 Using the .NET Client 301

About the .NET Data Provider 301

Specifying Connection Properties 302
Using Connection Pooling 302

Specifying Connection Attributes 305

Using .NET Objects 308

Assemblies 308

Parameter Markers 309

Parameter Arrays 309

Stored Procedures 310

Transaction Support 311
Using Local Transactions 311
Using Distributed Transactions 311

Tracing Method Calls 313

Connecting to a Database 314

10 Developing .NET Applications 319

Namespace 319

Data Types 320
Mapping Parameter Data Types 321

Isolation Levels 325

Threading 325

Event Handling 325

.NET Public Objects/Interfaces Supported 326
GetSchemaTable Method 330

.NET Data Provider Objects 333
SequeLinkCommand Object 334
SequeLinkCommandBuilder Object 336
SequeLinkConnection Object 337
SequeLinkDataAdapter Object 338
SequeLinkDataReader Object 338
SequeLinkError Object 339
SequeLinkErrorCollection Object 340
SequeLinkException Object 340
SequeLinkInfoMessageEventArgs Object 341
SequeLinkParameter Object 342
SequeLinkTrace Object 343
SequeLinkTransaction Object 344

Setting .NET Security Permissions 345
Code Access Permissions 345
Security Attributes 345

Error Handling 345
.NET Errors 346
SequeLink Client Errors 346
SequeLink Server Errors 346
Database Errors 347

Designing .NET Applications for Performance 347
Retrieving Data 348
Selecting .NET Objects and Methods 351
Designing .NET Applications 352
Updating Data 353

For More Information 354

Part 5: Reference

A SQL Escape Sequences 357

Date, Time, and Timestamp Escape Sequences 358

Scalar Functions 359
String Functions 366
Numeric Functions 368
Date and Time Functions 370
System Functions 373

Like Predicate Escape Characters 373

Outer Join Escape Sequences 374

B Data Types and Isolation Levels 377

Supported Data Types 377
DB2 on OS/390 378
DB2 UDB V7, V8 382
Informix 9 389
Microsoft SQL Server 7, 2000 393
Oracle8, Oracle9i 401
Sybase 407

Isolation Levels 413

C JDBC Support 415

JDBC Compatibility 415

Supported Functionality 415

D JDBC Connection Pool Manager 465

Creating a Data Source 465
Creating a DataDirect SequeLink Data Source Object 466
Creating a Data Source Using the DataDirect Connection Pool Manager 468

Connecting to a Data Source 471

Terminating the Pool Manager 474

E .NET Code Examples 475

Sample Tables Used in the Code Examples 475
Sample Tables for Oracle 475
Sample Tables for Sybase 477

Retrieving Data Using a DataReader 478

Using a Local Transaction With a DataReader 479

Using a Distributed Transaction 481

Using the CommandBuilder 483

Retrieving a Result Set using a DataAdapter Object 484

Limiting the Rows Returned by a Select Statement 486

Updating Data in a DataSet 487

Calling a Stored Procedure 489

Retrieving Warning Information 490

Retrieving a Scalar Value 491

Index 493

List of Tables

Table 1-1. ODBC Attributes 50

Table 2-1. Sources for Required ODBC Development Tools. 64

Table 2-2. Compiler Requirements for UNIX 65

Table 2-3. ODBC Function Conformance for 2.x ODBC Applications 66

Table 2-4. Function Conformance for 3.*x* ODBC Applications. 67

Table 2-5. Support for Unicode ODBC W (Wide) Function Calls 68

Table 2-6. Multithreading Functionality of the ODBC Driver 70

Table 2-7. Using SQLCancel in Multithreaded Applications 71

Table 2-8. Support for Scrollable Cursors (ODBC) 72

Table 3-1. DataDirect Technologies Configuration Manager: Parts and Functions for ADO Data Sources 108

Table 3-2. ADO Connection Attributes 130

Table 4-1. Objects and Interfaces Supported by the ADO Provider 136

Table 4-2. OLE DB Schema Rowsets Supported by the ADO Provider 138

Table 4-3. OLE DB Property Groups Supported by the ADO Provider 139

Table 4-4. OLE DB Data Source Property Supported by the ADO Provider 140

Table 4-5. OLE DB Data Source Information Properties Supported by the ADO Provider 141

Table 4-6. Initialization Properties Supported by the ADO Provider 146

Table 4-7. Rowset Properties Supported by the ADO Provider 148

Table 4-8. Session Properties Supported by the ADO Provider 152

Table 4-9. Supported OLE DB Interfaces Used by ADO 152

Table 4-10. Dynamic Properties Used for the ADO Command Object 154

Table 4-11. Mapping Methods Supported by the ADO Connection Object 157

Table 4-12. Dynamic Properties Supported for the ADO Connection Object . . . 158
Table 4-13. Mapping Methods Supported by the Recordset Object 165
Table 4-14. Dynamic Properties Used for the Recordset Object. 167
Table 5-1. JDBC Client Directory and Files. 184
Table 5-2. Support for the Java Transaction API (JTA) by the JDBC Client 195
Table 5-3. JDBC Properties 201
Table 8-1. Supported JDBC Features 272
Table 8-2. Using Cancel in Multithreaded JDBC Applications 275
Table 8-3. Support for Scrollable Cursors (JDBC). 277
Table 9-1. Connection String Attributes for the .NET Client 306
Table 10-1. Mapping of SequeLink Data Types 320
Table 10-2. Mapping of the System.Data.DbTypes to SequeLinkDbTypes 322
Table 10-3. Mapping .NET Framework Data Types to SequeLinkDbTypes 324
Table 10-4. Supported .NET Framework Objects/Interfaces for the .NET Data Provider 327
Table 10-5. Columns Returned by GetSchemaTable on DataReader. 330
Table 10-6. Properties of the SequeLinkCommand Object. 334
Table 10-7. Properties of the SequeLinkCommandBuilder Object. 336
Table 10-8. Properties of the SequeLinkConnection Object. 337
Table 10-9. Properties of the SequeLinkDataReader Object 339
Table 10-10. Properties of the SequeLinkError Object 339
Table 10-11. Properties of the SequeLinkException Object 341
Table 10-12. Properties of the SequeLinkInfoMessageEventArgs Object 342
Table 10-13. Properties of the SequeLinkParameter Object 342
Table 10-14. Properties of the SequeLinkTrace Object. 343
Table 10-15. Methods of the SequeLinkTransaction Object. 344
Table A-1. Language Features 357

Table A-2. Scalar Functions Supported on DB2 359
Table A-3. Scalar Functions Supported on Informix 362
Table A-4. Scalar Functions Supported on Microsoft SQL Server 363
Table A-5. Scalar Functions Supported on Oracle 364
Table A-6. Scalar Functions Supported on Sybase 365
Table A-7. Scalar String Functions 366
Table A-8. Scalar Numeric Functions 369
Table A-9. Scalar Time and Date Functions 371
Table A-10. Scalar System Functions 373
Table A-11. Outer Join Escape Sequences Supported 375
Table B-1. Mapping Data Types for DB2 on OS/390 to ODBC Data Types 378
Table B-2. Mapping Data Types for DB2 on OS/390 to ADO Data Types 379
Table B-3. Mapping Data Types for DB2 on OS/390 to JDBC Data Types 380
Table B-4. Mapping Data Types for DB2 on OS/390 to .NET Data Types 381
Table B-5. Mapping DB2 UDB Data Types to ODBC Data Types 383
Table B-6. Mapping DB2 UDB Data Types to ADO Data Types 385
Table B-7. Mapping DB2 UDB Data Types to JDBC Data Types 386
Table B-8. Mapping DB2 UDB Data Types to .NET Data Types 388
Table B-9. Mapping Informix 9 Data Types to ODBC Data Types 389
Table B-10. Mapping Informix 9 Data Types to ADO Data Types 390
Table B-11. Mapping Informix 9 Data Types to JDBC Data Types 391
Table B-12. Mapping Informix 9 Data Types to .NET Data Types 392
Table B-13. Mapping Microsoft SQL Server Data Types to ODBC Data Types . . . 393
Table B-14. Mapping Microsoft SQL Server Data Types to ADO Data Types 395
Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types 397
Table B-16. Mapping Microsoft SQL Server Data Types to .NET Data Types 399

Table B-17. Mapping Oracle Data Types to ODBC Data Types 401
Table B-18. Mapping Oracle Data Types to ADO Data Types. 403
Table B-19. Mapping Oracle Data Types to JDBC Data Types 404
Table B-20. Mapping Oracle Data Types to .NET Data Types 406
Table B-21. Mapping Sybase Data Types to ODBC Data Types 407
Table B-22. Mapping Sybase Data Types to ADO Data Types 409
Table B-23. Mapping Sybase Data Types to JDBC Data Types 410
Table B-24. Mapping Sybase Data Types to .NET Data Types 411
Table B-25. Isolation Levels. 413

Preface

This book is your guide to developing client applications for DataDirect® SequeLink® 5.4 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 client to server for the latest data access standards, including Open Database Connectivity (ODBC), JDBC, ActiveX Data Objects (ADO), and ADO.NET.

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 29 provides information about using ODBC applications with the SequeLink *for* ODBC Client.

- Chapter 2 “Developing ODBC Applications” on page 63 provides information about developing ODBC applications for SequeLink environments.

Part 2: Developing ADO Applications

- Chapter 3 “Using the ADO Client” on page 105 provides information about using ADO applications with the SequeLink *for* ADO Client.
- Chapter 4 “Developing ADO Applications” on page 135 provides information about developing ADO applications for SequeLink environments.

Part 3: Developing JDBC Applications

- Chapter 6 “Using DataDirect Test” on page 211 introduces DataDirect Test, 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 5 “Using the JDBC Client” on page 179 provides information about using JDBC applications with the SequeLink *for* JDBC Client.
- Chapter 7 “Tracking JDBC Calls” on page 263 introduces Spy, a development software component that allows you to track JDBC calls, and describes how to use it.
- Chapter 8 “Developing JDBC Applications” on page 271 provides information about developing JDBC applications for SequeLink environments.

Part 4: Developing .NET Applications

- Chapter 9 “Using the .NET Client” on page 301 provides information about using .NET applications with the SequeLink *for* .NET Client.
- Chapter 10 “Developing .NET Applications” on page 319 provides information about developing .NET applications for SequeLink environments.

Part 5: Appendixes

- Appendix A "SQL Escape Sequences" on page 357 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 377 lists the data types and isolation levels supported for each data store supported by SequeLink.
- Appendix C "JDBC Support" on page 415 provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- Appendix D "JDBC Connection Pool Manager" on page 465 provides sample code as an example of using the DataDirect Connection Pool Manager to allow your applications to handle connection pooling.
- Appendix E ".NET Code Examples" on page 475 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	*Getting Started with SequeLink*
Installing the SequeLink middleware components	*SequeLink Installation Guide*
Administering your SequeLink environment	*SequeLink Administrator's Guide*
Developing ODBC, ADO, JDBC, and .NET applications for the SequeLink environment	*SequeLink Developer's Reference*
Troubleshooting and referencing error messages	*SequeLink Troubleshooting Guide and Reference*

SequeLink documentation is provided on your DataDirect CD in PDF format, which allows you to view it online or print it. You can view the SequeLink online documentation using Adobe Acrobat Reader. The DataDirect CD includes Acrobat Reader 5.*x* with Search for Windows, and Acrobat Reader 4.*x* with Search for UNIX. SequeLink product documentation is also available on the DataDirect Technologies Web site:

http://www.datadirect-technologies.com/techres/proddoc_product.asp

On Windows and UNIX, you can choose to install the online books on your system. When installed, they are located in the books directory that is created beneath the SequeLink installation directory.

When you install the JDBC, .NET, or ODBC Client, HTML-based online help for developing applications is placed by default in the help directory that is created beneath the SequeLink installation directory. To access help, you must have Internet Explorer 5.*x* or higher, or Netscape 4.*x* or higher, installed. (Netscape 6.0. does not support the help system; however, 6.1 or higher does.) After you have opened the main screen of the help system in your browser (as described below), you can bookmark it in the browser for quick access later.

On Windows platforms, help is available from the setup dialog for the ODBC driver and ADO data provider. When you click **Help**, your browser will open to the correct topic. There are three navigation buttons at the left end of the grey menu bar at the top of the help screen. Clicking the left-hand button displays the table of contents and the index for the entire help system in the left pane. After the left pane is displayed, clicking the left-hand button synchronizes the contents of the right-hand pane with its location in the table of contents. The next two buttons navigate to the previous and following pages. The two buttons at the right end of the menu bar allow you print or bookmark the page. The program group for SequeLink also contains an icon for launching the help system.

There are three navigation buttons at the left end of the grey menu bar at the top of the help screen. Clicking the left-hand button synchronizes the contents of the right-hand pane with its location in the table of contents. The next two buttons navigate to the previous and following pages. The two buttons at the right end of the menu bar allow you print or bookmark the page.

Conventions Used in This Book

This section describes the typography, terminology, and other conventions used in this book.

Typographical Conventions

This book uses the following typographical conventions:

Convention	Explanation
italics	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 the name of a file. For operating environments that use case-sensitive filenames, the correct capitalization is used in information specific to those environments. Also indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
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: SELECT [DISTINCT], DISTINCT is an optional keyword.
braces { }	Indicates that you must select one item. For example, {yes | no} 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
	Windows. Information specific to the Microsoft Windows 98, Windows Me, Windows NT, Windows 2000, Windows Server 2003, and Windows XP environments is identified by the Windows symbol.
NT	*Windows NT*. Information specific to the Microsoft Windows NT environment is identified by the Windows symbol and the letters NT.
200*x*	*Windows 200x*. Information specific to the Microsoft Windows 2000 and Windows Server 2003 environment is identified by the Windows symbol and the characters 200*x*.

Symbol	Environment
Server	*Windows Server*. Information specific to the Microsoft Windows platforms on which SequeLink Server runs is identified by the Windows symbol and the word Server.
Client	*Windows Client*. Information specific to the Microsoft Windows platforms on which SequeLink Clients run is identified by the Windows symbol and the word Client.
XP	*Windows XP*. Information specific to the Microsoft Windows XP environment is identified by the Windows symbol and the letters XP.
UNIX	*UNIX*. Information specific to UNIX environments is identified by this symbol, which applies to all supported UNIX environments. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.
z/OS	*z/OS*. Information specific to OS/390 and z/OS environments is identified by the characters z/OS.

Contacting Technical Support

DataDirect Technologies provides technical support for registered users of this product, including limited installation support, for the first 30 days. Register online for your SupportLink user ID and password for access to the password-protected areas of the SupportLink web site at http://www.datadirect-technologies.com/support/support_index.asp. Your user ID and password are issued to you by email upon registration.

For post-installation support, contact us using one of the methods listed below or purchase further support by enrolling in

the SupportLink program. For more information about SupportLink, contact your sales representative.

The DataDirect Technologies web site provides the latest support information through SupportLink Online, our global service network providing access to support contact details, tools, and valuable information. Our SupportLink users access information using the web and automatic email notification. SupportLink Online includes a knowledge base so you can search on keywords for technical bulletins and other information.

World Wide Web

http://www.datadirect-technologies.com/support/support_index.asp

E-Mail

USA, Canada, and Mexico	supportlink@datadirect-technologies.com
Europe, Middle East, and Africa	int.supportlink@datadirect-technologies.com
Japan	jpn.answerline@datadirect.co.jp
All other countries	http://www.datadirect-technologies.com/contactus/distributor.asp provides a list of the correct e-mail contacts.

Local Telephone Support

Local phone numbers can be found at:

http://www.datadirect-technologies.com/support/support_contact_aline.asp

SupportLink support is available 24 hours a day, seven days a week.

Fax Information

Fax US, Mexico, and Canada	1 919 461 4527
Fax EMEA	+32 (0) 15 32 09 19

When you contact us, please provide the following information:

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

Part 1: Developing ODBC Applications

This part contains the following chapters:

- Chapter 1 "Using the ODBC Client" on page 29 provides information about using ODBC applications with the SequeLink *for* ODBC Client.
- Chapter 2 "Developing ODBC Applications" on page 63 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 *for* ODBC Client (the ODBC Client).

About the ODBC Client

The ODBC Client supports ODBC applications through a component called the *SequeLink for ODBC driver*. On Windows and UNIX 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 SequeLink *for* 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 for ODBC 5.4**, 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 UNIX" on page 46 for instructions on creating ODBC client data sources for UNIX.

Configuring ODBC Client Data Sources on Windows

To configure 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 for ODBC 5.4**, and then select the **ODBC Administrator** application.

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

2 To configure a new data source, click the **Add** button. A list of installed drivers appears. Select **DataDirect 32-BIT SequeLink 5.4**; 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.

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

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”.

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.

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

NOTE: We strongly recommend that you do not configure an ODBC translator and 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. Select a translator. When satisfied with your choice, click **OK** to close this dialog box and perform the translation.

NOTE FOR LDAP USERS: 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. If unspecified, the ODBC Client will use the default LDAP port 389.

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

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

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.

To configure ODBC file client data sources:

1. Start the ODBC Administrator by clicking **Start**, then **Programs**. From the Programs menu, select **DataDirect SequeLink for ODBC 5.4**, 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 32-BIT SequeLink 5.4**; 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 `UserLDAP=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've 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 39. 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.

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.

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 Logon to SequeLink service dialog box appears.

NOTE: If the password is not expired, the previously described 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 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 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.

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 for ODBC 5.4**, 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 Client Data Sources on Windows" on page 31, 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 UNIX

For 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 UNIX.

Configuring the System Information File

To configure an ODBC data source for 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 Attributes" on page 50.

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 32-BIT SequeLink 5.4

[SALESDB]
Driver=SALES/lib/ivslk19.so
Description=DataDirect 32-BIT SequeLink 5.4
Host=
Port=
UseLDAP=0
DistinguishedName=

[ODBC]
Trace=0
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 64-BIT SequeLink 5.4

[AccountingDB]
Driver=Accounting/lib64/ivslk19.so
Description=DataDirect 64-BIT SequeLink 5.4
Host=
Port=
UseLDAP=0
DistinguishedName=

[ODBC]
Trace=0
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 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` (for 32-bit client)
  - `. .sqlnk64.sh` (for 64-bit client)
- If you are using the C shell, type:
  - `source .sqlnk.csh` (for 32-bit client)
  - `source .sqlnk64.csh` (for 64-bit client)

Executing this shell script sets the following environment variables:

ODBCINI	Specifies where the centralized odbc.ini or odbc64.ini 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 UNIX is a multi-user environment, 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 a [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/slodbc54
```

Connecting Using a Connection String

If yourapplication 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.

You can specify long or short names in the connection string, which has the format:

```
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 example above.

See "ODBC Connection Attributes" on page 50 for a list of ODBC connection attributes and their valid values.

ODBC Connection Attributes

Table 1-1 "ODBC Attributes" on page 50 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 Attributes

Attribute	Description
ApplicationID (APPID)	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. See "Specifying Application IDs" on page 78 for more information about using application IDs to limit access to the SequeLink services.
ApplicationName (APPNAME)	Identifies the application that is establishing the connections. The initial default value is SequeLink for ODBC Application.

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

Attribute	Description
AutomaticApplicationID (AUTOAPPID)	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. See "Specifying Application IDs" on page 78 for more information about using application IDs to limit access to SequeLink services.
BlockFetchForUpdate (BFFU)	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. When set to 0, the appropriate row is locked. When set to 1, the appropriate row is not locked. The initial default value is 1. NOTE: Specifying 0 will degrade the performance for SELECT FOR UPDATE statements because rows will be fetched one at a time.
Database (DB)	Specifies the name of the database to which you want to connect.
DSN (DSN)	Specifies a string that identifies an ODBC data source configuration. Examples include “Accounting” or “SequeLink to Oracle Data”.
DBLogonID (DBUID)	Specifies the data store user name, which may be required depending on the server configuration.
DBPassword (DBPWD)	Specifies the data store password, which may be required depending on the server configuration.
DistinguishedName (DN)	Specifies the distinguished name identifying the LDAP entry from which connection information is retrieved. This attribute is required when UseLDAP=1.

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

Attribute	Description
EnableDescribeParam (EDP)	EnableDescribeParam={0 |1}. Specifies a workaround connection attribute for connections to Oracle data stores only. When set to 0, support is turned off for SQLDescribeParam. When set to 1, support is turned on for SQLDescribeParam and will describe all parameters as SQL_CHAR with a precision of 999. The initial default value is 0.
FetchNextOnly (FNO)	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). 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. When set to TRUE, the driver will incorrectly report that only SQL_FETCH_NEXT is supported, which satisfies RDO. When set to FALSE, the driver will correctly report other values than SQL_FETCH_NEXT. The initial default value is FALSE.

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

Attribute	Description
FixCharTrim (FCT)	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. When set to 0, the workaround is turned off. When set to 1, SQL_CHAR data that is not padded with spaces is returned. The initial default value is 0.
GetOutputParams (GOP)	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: When set to 1, output parameters are returned after an execute. When set to 2, output parameters are returned after a fetch is complete. When set to 4, output parameters are returned after moreresults returns no more rows. When set to 7, output parameters are returned after all of the above. The value for this connection attribute should be set to the cumulative value of all chosen options added together. The initial default value is 7. NOTE: Set GetOutputParams=3 when executing stored procedures with output parameters in RDO (Visual Basic 5 and 6).
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.

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

Attribute	Description
Host (HST)	Specifies the TCP/IP address of the SequeLink Server, specified in dotted format or as a host name. 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.
LogonID (UID)	Specifies the host or data store user name, which may be required depending on the server configuration.
NewPassword (NPWD)	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: `[DataDirect][ODBC SequeLink driver][SequeLink Server] The user password was changed successfully` If unspecified and the SequeLink Server detects that the host password has expired, you will be prompted for a new host password. For more information about the SequeLink password change mechanism, refer to the *SequeLink Administrator's Guide*.
Password (PWD)	Specifies the host or data store password, which may be required depending on the server configuration.

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

Attribute	Description
Port (PRT)	Specifies the TCP/IP port on which the SequeLink Server is listening. 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.
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)	Imposes a time limit on: ■ 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).
SLKStaticCursorLongColBuffLen (SSCLCBL)	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. 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. The initial default value is 4.

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

Attribute	Description
UseLDAP (LDAP)	UseLDAP={0 | 1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP. When set to 0, the SequeLink Client will connect directly to the specified SequeLink Server. 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. The initial default value is 0.
WorkArounds (WA)	Turns on workarounds that allow you to take full advantage of the ODBC driver with ODBC applications requiring nonstandard or extended behavior. 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. 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. When set to 2, the driver reports that qualifiers are not supported. This option is provided because some applications cannot handle database qualifiers.

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

Attribute	Description
WorkArounds *(cont.)*	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. 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. 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 `tablename is not a valid name` error. When set to 32, users of flat-file drivers are allowed to abort a long-running query by pressing the ESC key. When set to 64, the result is a column name of C*position* where *position* is the ordinal position in the result set. For example: `SELECT col1, col2+col3 FROM table1` produces the column names col1 and C2. 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. When set to 256, SQLGetInfo/ SQL_ACTIVE_CONNECTIONS is forced to return as 1. When set to 512, the SQLSpecialColumns function returns a unique index as returned from SQLStatistics to prevent ROWID results.

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

Attribute	Description
WorkArounds *(cont.)*	When set to 2048, SQLDriverConnect returns "Database='' instead of "DB='' in the returned connection string. 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. 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 = ". When set to 524288, the precision and scale settings for SQL_DECIMAL parameters are overridden to precision 40 and scale 20. 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. When set to 536870912, SQLBindParameter is allowed to be called after SQLExecute to change the precision of previously bound parameters. 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.

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

Attribute	Description
WorkArounds2 (WA2)	Turns on workarounds that allow you to take full advantage of the ODBC driver with ODBC applications requiring nonstandard or extended behavior. 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. 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. 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 data store. When set to 32, Microsoft Access requires that the characters "DSN=" are returned by SQLDriverConnect in the connection string output parameter.

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”.

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 initial default value 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.

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

NOTE: We strongly recommend that you do not configure an ODBC translator and 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. Select a translator. When satisfied with your choice, click **OK** to close this dialog box and perform the translation.

NOTE FOR LDAP USERS: 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. If unspecified, the ODBC Client will use the default LDAP port 389.

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

2 Developing ODBC Applications

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

- "Required ODBC Libraries and Header Files" on page 64
- "UNIX Compiler Requirements" on page 65
- "ODBC API Functions" on page 65
- "SQL Escape Sequences" on page 68
- "Data Types and Isolation Levels" on page 69
- "Threading" on page 69
- "Using Scrollable Cursors" on page 72
- "Using Stored Procedures with Oracle" on page 74
- "Specifying Application IDs" on page 78
- "Persisting a Result Set as an XML Data File" on page 80
- "Error Handling" on page 82
- "Developing Performance-Optimized ODBC Applications" on page 84

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.

UNIX Compiler Requirements

The ODBC Client has specific compiler requirements on UNIX. Applications must be compiled using the guidelines shown in Table 2-2.

Table 2-2. Compiler Requirements for UNIX

UNIX Platform	Compiler Requirements
AIX reentrant	■ 32-bit compiler version: C Set++ for AIX 3.6.4.0 (using the reentrant version, xlC_r) or higher ■ 64-bit compiler version: C for AIX Compiler 5.0.2.0. ■ 64-bit Compiler flag: ODBC64 (AIX 5.*x* only)
Solaris	SUNWorkshop (Forte C++) 6.0 or higher
Linux	GNU project C++ Compiler version 2.96 64-bit Compiler flag: ODBC64
HP-UX 11 aCC	HP aC++ Compiler version A.03.05 or higher

ODBC API Functions

The ODBC driver is ODBC Level 1–compliant, supporting all ODBC Core and Level 1 functions. Most Level 2 functions are also supported. Table 2-3 and Table 2-4 list supported functions for ODBC 2.*x* and ODBC 3.*x* applications, respectively.

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

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 2-4. 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[1]
SQLDescribeParam[Note]	SQLGetFunctions	SQLSetStmtAttr
SQLDisconnect	SQLGetInfo	SQLSpecialColumns
SQLDriverConnect	SQLGetStmtAttr	SQLStatistics
SQLDrivers	SQLGetTypeInfo	SQLTablePrivileges
SQLEndTran	SQLMoreResults	SQLTables
SQLExecDirect	SQLNativeSql	SQLTransact

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

Support for Unicode ODBC W (Wide) Function Calls

The ODBC driver fully supports the SQL-W functions.

On both Windows and UNIX, the ODBC driver automatically determines whether the database is a Unicode database. Table 2-5 shows how the ODBC driver maps the database data types to the Unicode data types.

Table 2-5. 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.

2. On the SequeLink Server for ODBC Socket, the ODBC driver supports the data types of the backend database.

3. The ODBC driver does not support Unicode data types with SequeLink Server for Informix.

SQL Escape Sequences

See Appendix A "SQL Escape Sequences" on page 357 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 377 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 multithreading ability of the ODBC driver is platform dependent as shown in Table 2-6.

Table 2-6. Multithreading Functionality of the ODBC Driver

Platform	Capability
Windows	Thread for each connection
Solaris	Thread for each connection
AIX reentrant	Thread for each connection
Linux	Thread for each connection
HP-UX 11 aCC	Thread for each connection using the HP-UX native threading model (p-threads)

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 will return `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 V5, V6, V7R1 on OS/390	Ignored
DB2 V7.2 and V8.1 on Windows	OK
DB2 V7.2 and V8.1 on UNIX	Ignored
Informix 9.*x*, Informix 2000	OK
Microsoft SQL Server 7.0, 2000	OK
Oracle8, Oracle9i on Windows NT, Windows 2000, Windows XP	Ignored
Oracle8, Oracle9i on UNIX	OK
Sybase 11, 12	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.

The type of scrollable cursors that can be used depend on the data store to which you are connecting. Table 2-8 shows the type of scrollable cursors supported for each database.

Table 2-8. Support for Scrollable Cursors (ODBC)

Database	Static	Keyset-Driven
DB2 on OS/390	X	
DB2 UDB on Windows	X	
Informix	X	X
Microsoft SQL Server (see note)	X	X
ODBC Socket	X	
Oracle8 and Oracle9	X	X
Sybase (see note)	X	X

NOTE: To use keyset-driven cursors with Microsoft SQL Server and 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 Attributes," on page 50 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 80 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=EMPLOYEEINFOPROC,
...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", "EMPLOYEEINFOPROC", "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 either 1, 2, or 3.

- **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 either 1, 2, or 3.

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
StatementHandle	The handle of the statement that contains the result set to persist as XML.

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

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

```
Function Sequence Error
```

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 84
- "Retrieving Data" on page 90
- "ODBC Function Selection" on page 94
- "Designing ODBC Applications" on page 97
- "Updating Data" on page 99

Catalog Functions

The following ODBC functions are catalog functions:

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

Minimizing the Use of Catalog Functions

Compared to other ODBC functions, catalog functions are relatively slow. By caching information returned from catalog functions, applications can avoid multiple executions.

While almost no ODBC application can be written without using catalog functions, you can improve performance by minimizing their use. To return all result column information mandated by the ODBC specification, an ODBC 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 expenses.

Applications should cache information from catalog functions.

Avoiding Search Patterns

Passing null arguments to catalog functions generates time-consuming queries. In addition, network traffic may increase because of unwanted result set information. Always supply as many non-null arguments to catalog functions as possible.

Because catalog functions are slow, applications should invoke them efficiently. Any information that the application can send the driver when calling catalog functions can result in improved performance and reliability. However, many applications pass the minimum number of non-null arguments required for the function to return a successful result set.

For example, consider a call to SQLTables where the application requests information about the table “Customers.” Often, this call is coded as shown:

```
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 unnecessary rows would be filtered by the database.

In addition, if the TableType argument can be supplied, the SQL sent to the server can be changed 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 columns. 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, consider a DBMS server that does not support natively preparing a SQL statement. The performance of Case 1 would not change, but the performance of Case 2 would improve slightly, because the dummy query must be 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 performs better than 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, OLE DB, and JDBC 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, OLE DB, and JDBC 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 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

This section provides general guidelines for 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, the application developer should 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 a manageable limit by calling SQLSetStmtOption with the SQL_MAX_LENGTH option. 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, the driver optimizes its network use by retrieving long data in large chunks and returning it to the user when requested.

For example:

```
char CaseContainer[1000];
...
rc = SQLExecDirect (hstmt, "SELECT CaseHistory FROM Cases
   WHERE CaseNo = 71164", SQL_NTS);
...
rc = SQLFetch (hstmt);
rc = SQLGetData (hstmt, 1, CaseContainer,(SWORD)
   sizeof(CaseContainer), ...);
```

At this point, it is more likely that an ODBC driver will retrieve 64 KB of information from the server, rather than 1000 bytes. One 64-KB retrieval requires less network traffic than 64 1000-byte retrievals. Unfortunately, the application might not call SQLGetData again; thus, the first and only retrieval of CaseHistory would be slowed by the fact that 64 KB of data had to be sent across the network.

Many ODBC drivers allow you to limit the amount of data retrieved across the network by using the statement attribute SQL_MAX_LENGTH. This attribute allows the driver to communicate to the database server that only *Z* bytes of data are relevant to the client. The server responds by sending only the first Z bytes of data for all result columns. This optimization substantially reduces network traffic and improves performance. Our 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. 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 ODBC 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 ODBC driver retrieving information into a container and copying that information to the user's buffer, the ODBC 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 becomes less complex, resulting in more easily maintainable code.

Most ODBC drivers 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));
```

The number of ODBC calls made by the application is reduced from 110 to 4 (1 SQLSetStmtOption + 1 SQLExecDirect + 1 SQLBindCol + 1 SQLExtendedFetch). Note the total savings from an initial call load of more than 1890 ODBC calls to 4 ODBC calls. 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 your application's 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).

ODBC Function Selection

The guidelines in this section help you to optimize system performance when selecting and using ODBC functions.

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 a statement that uses parameter markers. SQLExecDirect is optimized for a single execution of a SQL statement. Unfortunately, more than 75 percent 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 has substantial overhead, but the statement will be executed multiple times. Although creating stored procedures is performance-expensive, processing is minimal because the query is parsed and optimization paths are stored when the procedure is created.

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

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 scrollable cursors are provided by the driver, 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 provides 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 one of the slowest operations on PCs. 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 this ability 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);
```

Designing ODBC Applications

This general guidelines in this section will help you to optimize system performance when designing 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. Many ODBC applications contain poorly designed elements for connection management. Avoid connecting to a data source after establishing an initial connection.

Some ODBC applications are designed to call informational gathering routines that have no record of attached connection handles. For example, some applications establish a connection and call a routine in a separate DLL or shared library that reattaches and gathers information about the driver.

Although gathering ODBC driver information at connection is a good practice, it often is more efficient to gather it in one-step rather than two steps. One popular commercial ODBC

application connects a second time to gather driver information, and *never* disconnects the second connection. Applications that are designed as separate entities should pass the 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 perform SQL statements. Applications should use *statement handles* to manage multiple SQL statements.

In Windows, significant performance improvement can be achieved 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 Transactions

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

What does a commit 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.

Furthermore, some database servers do not provide an Autocommit mode. For this server type, 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.

Updating Data

This section provides general guidelines for updating data in databases that will help optimize system performance.

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 attempt to 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 developer 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 usually update data by using 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 usually provide the fastest access to the data, because they usually 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 105 provides information about using ADO applications with the SequeLink *for* ADO Client.
- Chapter 4 “Developing ADO Applications” on page 135 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 *for* ADO Client (the ADO Client).

About the ADO Client

The ADO Client supports ADO/OLE DB applications through a component called the *SequeLink for ADO 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 111 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 121 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 for ADO 5.4**. Then, select the **DataDirect Configuration Manager** application.

The Configuration Manager window is divided into two panes. As Figure 3-2 on page 110 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\slado54 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
		■ Modify a data source
	View	■ View or hide the toolbar and status bar
		■ Refresh the Configuration Manager
	Tools	■ 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.

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...
Vertical splitter bar	Adjust the size of the left and right panes.
Status bar	■ 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 for ADO 5.4**. 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 SequeLink 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 112
- "Modifying an ADO Client Data Source" on page 115
- "Renaming an ADO Client Data Source" on page 115
- "Deleting an ADO Client Data Source" on page 116
- "Copying an ADO Client Data Source" on page 116
- "Changing Data Source Directories" on page 117
- "Defining Default Setup Options" on page 118

Creating an ADO Client Data Source

1 Start the DataDirect Configuration Manager. To start the Configuration Manager, select **Start** / **Programs**, and select **SequeLink for ADO Client 5.4**. 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 5.4 for ADO Provider**.

5 Click the **Set Up Data Source** button. The DataDirect SequeLink for 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.

NOTE FOR LDAP USERS: 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. If unspecified, the ADO Client will use the default LDAP port 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*.

NOTE: All data sources are saved to the current directory displayed in the Configuration Manager. See "Changing Data Source Directories" on page 117 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 112 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 for ADO 5.4**. 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 for ADO 5.4**. 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 128 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 122 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 window 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.

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 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 Logon to SequeLink Service dialog box appears.

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.

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 Technologies 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 Technologies 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;"
```

See "ADO Connection Attributes" on page 130 for a list of 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 Table 3-2 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
Application ID	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. See "Specifying Application IDs" on page 173 for more information about using application IDs to limit access to SequeLink services.
ApplicationName	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. The initial default value is SequeLink for ADO Application.
Automatic Application ID	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. See "Specifying Application IDs" on page 173 for more information about using application IDs to limit access to SequeLink services.

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

Attribute	Description
Database	Specifies the name of the database to which you want to connect.
Database User Name	Specifies the data store user name, which may be required depending on the server configuration.
Database Password	Specifies the data store password, which may be required depending on the server configuration.
Data Source	Specifies a string that identifies an ADO/OLE DB data source configuration. Examples include “Accounting” or “SequeLink to Oracle Data.”
Default Length for Long Data	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. The initial default value is 4.
Distinguished Name	Specifies the distinguished name identifying the LDAP entry from which connection information is retrieved. This attribute is required when UseLDAP=1.
Host	Specifies the TCP/IP address of the SequeLink Server, specified in dotted format or as a host name. 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.
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.

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

Attribute	Description
New Password	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: `[DataDirect][SequeLink ADO Provider] [SequeLink Server] The user password was changed successfully` If unspecified and the SequeLink Server detects that the host password has expired, you will be prompted for a new host password. For more information about the SequeLink password change mechanism, refer to the *SequeLink Administrator's Guide*.
Password	Specifies the host or data store password, which may be required depending on the server configuration.
Port	Specifies the TCP/IP port on which the SequeLink Server is listening. 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.
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	UseLDAP={0 | 1}. Determines whether the parameters to establish a connection to the SequeLink Server should be retrieved from LDAP. When set to 0, the SequeLink Client will connect directly to the specified SequeLink Server. 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. The initial default value is 0.
User ID	Specifies the host or data store user name, which may be required depending on the server configuration.

4 Developing ADO Applications

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

- "OLE DB Objects and Interfaces" on page 136
- "Supported Schema Rowsets" on page 138
- "Supported OLE DB Property Groups" on page 139
- "OLE DB Interfaces Supported in ADO" on page 152
- "Mapping ADO Methods and Properties" on page 154
- "Data Shaping" on page 170
- "Persisting Information" on page 171
- "Using Rowsets" on page 171
- "Unicode Support" on page 172
- "Mapping Data Types" on page 172
- "Specifying Application IDs" on page 173
- "Error Handling" on page 174

OLE DB Objects and Interfaces

The ADO 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 provider supports, and the interfaces implemented for each object.

Table 4-1. Objects and Interfaces Supported by the ADO 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	
Eumerator	IDBInitialize	
	IDBProperties	
	IParseDisplayName	
	ISourcesRowset	
	ISupportErrorInfo	
MultipleResults	IMultipleResults	
	ISupportErrorInfo	

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

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

CATALOGS [1], [2]	PROCEDURE_PARAMETERS
COLUMN_PRIVILEGES[1], [2]	PROCEDURES
COLUMNS	PROVIDER_TYPES
FOREIGN_KEYS	SCHEMATA[1], [2]
INDEXES	STATISTICS
PRIMARY_KEYS	TABLE_PRIVILEGES[1], [2]
PROCEDURE_COLUMNS [1], [2]	TABLES

NOTES:

[1] Not supported for DB2
[2] Not supported for Oracle

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 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 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 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 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_ASYNCTXNABORT	VALUE=VARIANT_FALSE. The data provider cannot abort transactions asynchronously.
DBPROP_ASYNCTXNCOMMIT	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 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 105 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 the 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 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 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_OPENROWSETSUPPORT	VALUE=DBPROPVAL_ORS_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 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 the provider, for example, SequeLink for ADO Provider.
DBPROP_PROVIDERNAME	VALUE=`slslk`*`nn`*`.DLL` is the default, where *nn* is the release level of the 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=VARIANT_TRUE. 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=DBPROPVAL_SS_ISEQUENTIALSTREAM.

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

Property ID	Default Value and Description
DBPROP_ SUBQUERIES	A combination of zero or one or more of the following: VALUE=DBPROPVAL_SQ_COMPARISON VALUE= DBPROPVAL_SQ_CORRELATEDSUBQUERIES. This indicates that all predicates that support subqueries support correlated subqueries. VALUE=DBPROPVAL_SQ_EXISTS VALUE=DBPROPVAL_SQ_IN VALUE=DBPROPVAL_SQ_QUANTIFIED
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	Specifies the name the data source uses for a table. VALUE="Table"
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 Provider

Property Name	Default Value and Description
DBPROP_AUTH_ PASSWORD	Specifies the password to be used for connecting to the data source or enumerator. This corresponds to the Password connection attribute. See "ADO Connection Attributes" on page 130 for more information about connection attributes.
DBPROP_AUTH_PERSIST_SENSITIVE_AUTHINFO	VALUE=VARIANT_FALSE. The data source object cannot store the password or other sensitive authentication information.
DBPROP_AUTH_ USERID	Specifies the User ID to be used for connecting to the data source. This corresponds to the User ID connection attribute. See "ADO Connection Attributes" on page 130 for more information about connection attributes,.
DBPROP_INIT_CATALOG	Specifies the name of the initial or default catalog to use when connecting to the data source.
DBPROP_INIT_ DATASOURCE	Specifies the name of the data source or enumerator to which to connect.
DBPROP_INIT_ HWND	Specifies the window handle to use if the data source object or enumerator needs to prompt for additional information.
DBPROP_INIT_LCID	Specifies the preferred locale ID for the consumer.
DBPROP_INIT_ MODE	VALUE=DB_MODE_READ. The default is read-only.
DBPROP_INIT_OLEDBSERVICES	VALUE=0. The provider does not enable the OLE DB services.
DBPROP_INIT_PROMPT	VALUE=DBPROMPT_COMPLETE. Display a connection dialog only if missing information is needed.

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

Property Name	Default Value and Description
DBPROP_INIT_ PROVIDERSTRING	Specifies a provider-specific string that contains extra initialization information. See "Connecting with a Provider String" on page 128 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 130.

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

Table 4-7 "Rowset Properties Supported by the ADO Provider" on page 148 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 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_CANSCROLLBACKWARDS	VALUE=VARIANT_FALSE. IRowsOffset must be non-negative.
DBPROP_CHANGEINSERTEDROWS	VALUE=VARIANT_FALSE. DeleteRows returns a status of DBROWSTATUS_E_NEWLYINSERTED for newly inserted rows, and SetData returns DB_E_NEWLYINSERTED.

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

Property Name	Default Value and Description
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.
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.

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

Property Name	Default Value and Description
DBPROP_LOCKMODE	VALUE=DBPROPVAL_LM_NONE. The 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 provider supports reentrancy during callbacks to the IRowsetNotify interface.
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 the provider cannot detect whether it affects multiple rows.
DBPROP_RETURNPENDINGINSERTS	VALUE=VARIANT_FALSE. The methods that fetch rows cannot return pending insert rows.

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

Property Name	Default Value and Description
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 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	A combination of zero or one or more of the following: Value=DBPROPVAL_UP_CHANGE. SetData is supported. Value=DBPROPVAL_UP_DELETE. DeleteRows is supported. Value=DBPROPVAL_UP_INSERT. InsertRow is supported.

Session Property Group

Table 4-8 lists the properties the ADO 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 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 provider supports additional OLE DB interfaces that are not used by ADO.

See "OLE DB Objects and Interfaces" on page 136 for a description of the OLE DB objects and interfaces supported by the ADO 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
ICommandPrepare	If available

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

OLE DB Interface	Use by ADO
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 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 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 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 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 provider uses the free-threaded model.
Scroll Backward	VALUE=False. IRowsOffset must be non-negative.
Server Cursor	VALUE=False. The 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	Specifies the transaction isolation level while in auto-commit mode. A combination of zero or one or more of the following: VALUE=DBPROPVAL_TI_BROWSE VALUE=DBPROPVAL_TI_CURSORSTABILITY VALUE=DBPROPVAL_TI_ISOLATED VALUE=DBPROPVAL_TI_READCOMMITTED VALUE=DBPROPVAL_TI_READUNCOMMITTED VALUE=DBPROPVAL_TI_REPEATABLEREAD VALUE=DBPROPVAL_TI_SERIALIZABLE
Catalog Location	The value depends on the SequeLink service you are using. Possible values are: VALUE=1. The catalog name is at the start of the fully qualified name. VALUE=2. The catalog name is at the end of the fully qualified name.
Catalog Term	Specifies the name the data source uses for a catalog. VALUE=Database

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

ADO Property	Default Value and Description
Catalog Usage	Specifies how catalog names can be used in text commands. A combination of zero or one or more of the following: VALUE=1. Catalog names are supported in all Data Manipulation Language statements. VALUE=2. Catalog names are supported in all table definition statements. VALUE=4. Catalog names are supported in all index definition statements. VALUE=8. Catalog names are supported in all privilege definition statements.
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 provider.
DBMS Version	Specifies the version of the product that the provider is currently accessing, where *xx* is the specific version of the product.

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

ADO Property	Default Value and Description
Extended Properties	A provider-specific string that contains extra initialization information. Consumers should use this property only for provider-specific connection information. See "Connecting with a Provider String" on page 128 for information on using the provider string with the ADO provider.
GROUP BY Support	This value depends on the SequeLink service you are using. VALUE=1. The GROUP BY clause is not supported. VALUE=2. The GROUP BY clause must contain all nonaggregated columns in the select list. It cannot contain any other columns. 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. 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.
Heterogeneous Table Support	VALUE=0. The provider cannot join tables from different catalogs or providers.
Identifier Case Sensitivity	This value depends on the SequeLink service you are using. VALUE=1. Identifiers in SQL are case-sensitive and are stored in upper case in the system catalog. VALUE=2. Identifiers in SQL are case-insensitive and are stored in lower case in the system catalog. VALUE=4. Identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog. VALUE=8. Identifiers in SQL are case-insensitive and are stored in mixed case in the system catalog.
Initial Catalog	Specifies the name of the initial, or default, catalog to use when connecting to the data source.

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

ADO Property	Default Value and Description
Isolation Levels	This value depends on the SequeLink service you are using. One or a combination of zero or one or more of the following: VALUE=DBPROPVAL_TI_BROWSE VALUE=DBPROPVAL_TI_CHAOS VALUE=DBPROPVAL_TI_CURSORSTABILITY VALUE=DBPROPVAL_TI_ISOLATED VALUE=DBPROPVAL_TI_READCOMMITTED VALUE=DBPROPVAL_TI_READUNCOMMITTED VALUE=DBPROPVAL_TI_REPEATABLEREAD VALUE=DBPROPVAL_TI_SERIALIZABLE
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 provider must spawn multiple connections to support concurrent command, session, and rowset objects.
Multiple Parameter Sets	VALUE=True. The 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 provider supports multiple results objects.
Multiple Storage Objects	VALUE=True. The 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 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 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 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=slslk*nn*.DLL, where *nn* is the release level of the provider.
Provider Version	Specifies the version of the DataDirect data provider. Refer to the README for the version number.
Quoted Identifier Sensitivity	This value depends on the SequeLink service you are using. VALUE=1. Quoted identifiers in SQL are case-sensitive and are stored in upper case in the system catalog. VALUE=2. Quoted identifiers in SQL are case-insensitive and are stored in lower case in the system catalog. VALUE=4. Quoted identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog. VALUE=8. Quoted identifiers in SQL are case-insensitive and are stored in mixed case.
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	This value depends on the SequeLink service you are using. A combination of the following: VALUE=DBPROPVAL_SU_DML_STATEMENTS. Schema names are supported in all Data Manipulation Language statements. VALUE=DBPROPVAL_SU_TABLE_DEFINITION. Schema names are supported in all table definition statements. VALUE=DBPROPVAL_SU_INDEX_DEFINITION. Schema names are supported in all index definition statements. VALUE=DBPROPVAL_SU_PRIVILEGE_DEFINITION. Schema names are supported in all privilege definition statements.

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 provider connects through "friendly" data source names, this can be the actual name of the server.
SQL Support	Specifies the level of SQL grammar that the provider supports. The effects are cumulative. A combination of zero or one or more of the following: VALUE=DBPROPVAL_SQL_NONE VALUE=DBPROPVAL_SQL_ODBC_CORE VALUE=DBPROPVAL_SQL_ODBC_MINIMUM VALUE=DBPROPVAL_SQL_ODBC_EXTENDED VALUE=DBPROPVAL_SQL_ESCAPECLAUSES VALUE=DBPROPVAL_SQL_ANSI92_ENTRY VALUE=DBPROPVAL_SQL_ANSI92_ENTRY VALUE=DBPROPVAL_SQL_FIPS_TRANSITIONAL VALUE=DBPROPVAL_SQL_ANSI92_INTERMEDIATE VALUE=DBPROPVAL_SQL_ANSI92_FULL VALUE=DBPROPVAL_SQL_ANSI89_IEF VALUE=DBPROPVAL_SQL_SUBMINIMUM
Structured Storage	VALUE=1. The provider supports DBPROPVAL_SS_ISEQUENTIALSTREAM.

Subquery Support	Specifies the predicates in text commands that support subqueries. A combination of zero or one or more of the following:
	`VALUE=DBPROPVAL_SQ_CORRELATEDSUBQUERIES`
	`VALUE=DBPROPVAL_SQ_COMPARISON`
	`VALUE=DBPROPVAL_SQ_EXISTS`
	`VALUE=DBPROPVAL_SQ_IN`
	`VALUE=DBPROPVAL_SQ_QUANTIFIED`

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 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 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 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 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 provider supports the free-threaded model.
Scroll Backward	VALUE=False. IRowsOffset must be non-negative.
Server Cursor	VALUE=False. The 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 provider in the Connection object connect string by typing `Data Provider=DataDirect SequeLink for ADO Provider`. The 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 provider supports the IOpenRowset interface, which retrieves all data from a table for a consumer. In addition, the 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 138 for information on the schema rowsets supported.

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

Unicode Support

The ADO provider supports Unicode.

When the string is data, the ADO provider returns the string to the data consumer as ANSI strings. If the data consumer requests a different data type through the bindings, the ADO provider performs the appropriate conversion.

When the string is not data, the ADO provider converts the string to Unicode format before returning it to the data consumer. This is required to conform to the OLE DB specification.

Mapping Data Types

See Appendix B "Data Types and Isolation Levels" on page 377 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 *`x`* is either 1, 2, or 3.

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 6 “Using DataDirect Test” on page 211 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 5 “Using the JDBC Client” on page 179 provides information about using JDBC applications with the SequeLink *for* JDBC Client.
- Chapter 7 “Tracking JDBC Calls” on page 263 introduces Spy, a tool that allows you to track JDBC calls, and describes how to use it.
- Chapter 8 “Developing JDBC Applications” on page 271 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 *for* JDBC Client (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
- DataDirect Test
- DataDirect Connection Pool Manager
- J2EE Connector Architecture (JCA) resource adapters

JDBC Driver

The JDBC driver is compliant with the JDBC 3.0 specification. The JDBC driver also supports the JDBC 2.0 Optional Package, which provides the following functionality:

- Java Naming Directory Interface (JNDI) for naming databases
- Connection Pooling
- Distributed Transactions Management support

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 Secure Socket Layer (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

Spy is a development software component for tracking JDBC calls. Spy passes calls issued by an application to an underlying JDBC driver and logs detailed information about those calls. It provides the following advantages:

- Spy can only be used with the SequeLink *for* JDBC Driver and the DataDirect Technologies Connect *for* JDBC driver.
- 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, using the SequeLinkDataSource object.

DataDirect Test

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 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 465 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 (JCA) Resource Adapter" on page 196 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 PDF 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/slssl.jar	JAR file required for pre-JDK 1.4 JVMs. The files contain all classes of the JDBC driver that implement Secure Socket Layer (SSL) encryption.
driver/lib/slssl14.jar driver/lib/iaik_jce_full.jar	JAR files required for JDK 1.4 JVMs. The files contain all classes of the JDBC driver that implement Secure Socket Layer (SSL) encryption.

Table 5-1. JDBC Client Directory and Files *(cont.)*

Directories and Files	Description
driver/lib/sljc.rar	Resource Archive for use with J2EE and JCA Connectors.
help/*.*	Files for the HTML-based online help for the JDBC driver.
testforjdbc/lib/testforjdbc.jar	JAR file containing all the DataDirect Test classes. To use DataDirect Test, add this path to your CLASSPATH variable.
testforjdbc/testforjdbc.bat	Batch file that starts DataDirect Test.
testforjdbc/testforjdbc14.bat	Batch file that starts DataDirect Test (for use with JVM 1.4).
testforjdbc/testforjdbc.sh	UNIX shell script that starts DataDirect Test.
testforjdbc/testforjdbc14.sh	UNIX shell script that starts DataDirect Test (for use with JVM 1.4).
pool/lib/pool.jar	JAR file containing all the classes for the DataDirect Connection Pool Manager.
proxy/cmdsrvc.exe	Executable that registers the SequeLink Proxy Server as a Windows service.
proxy/decrypt.bat proxy/decrypt14.bat proxy/encrypt.bat proxy/encrypt14.bat	Batch files that decrypt and encrypt the private key of the SequeLink Proxy Server certificate.
proxy/decrypt.sh proxy/encrypt.sh	UNIX shell scripts that decrypt and encrypt the private key of the SequeLink Proxy Server certificate.
proxy/proxyserver.bat proxy/proxyserver14.bat	Batch file that starts the SequeLink Proxy Server.

Table 5-1. JDBC Client Directory and Files *(cont.)*

Directories and Files	Description
proxy/proxyserver.sh	UNIX shell script that starts the SequeLink Proxy Server.
proxy/cert/	Directory containingdemo 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.
spy/lib/spy.jar	JAR file containing all Spy classes. To use 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.

Table 5-1. JDBC Client Directory and Files (cont.)

Directories and Files	Description
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.

Loading 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 sql.drivers using the Java -D option. The sql.drivers property is defined as a colon-separated list of driver class names. For example:

  ```
  com.ddtek.jdbc.sequelink.SequeLinkDriver:
  sun.jdbc.odbc.JdbcOdbcDriver
  ```

  The sql.drivers property can be set like other Java properties, using the -D option. For example:

  ```
  java -Dsql.drivers=
  com.ddtek.jdbc.sequelink.SequeLinkDriver
  ```

- *Method 2*: Set the Java property sql.drivers from within your Java application or applet. To do this, code the following

lines in your JDBC application, and call DriverManager.getConnection():

```
Properties p = System.getProperties();
p.put ("sql.drivers",
"com.ddtek.jdbc.sequelink.SequeLinkDriver");
System.setProperties (p);
```

- *Method 3*: Explicitly load the driver class using the standard Class.forName() method. To do this, code the following lines and call DriverManager.getConnection():

```
Class.forName("com.ddtek.jdbc.sequelink.
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*.

If not using SSL encryption, the connection URL format is:

`jdbc:sequelink://`*`hostname`*`:`*`port`*`[;`*`key`*`=`*`value`*`]...`

If using SSL encryption, the connection URL format is:

`jdbc:sequelink:ssl://`*`hostname`*`:`*`port`*`[;`*`key`*`=`*`value`*`]...`

where:

hostname	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 *SequeLink Administrator's Guide*.

port is the TCP/IP port on which the SequeLink server is listening. A default installation of SequeLink Server uses the port 19996.

key=value specifies connection properties. See "JDBC Connection Properties" on page 201 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 191, 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.

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 191 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 192 and "Example B: With Connection Pooling" on page 193.

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 191 for instructions on creating JDBC data sources. See Appendix D "JDBC Connection Pool Manager" on page 465 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?
Oracle8, Oracle9i	Yes
Informix 9	Yes
DB2 on OS/390 V5, V6, V7R2	Yes
DB2 UDB 7.2, V8	Yes
ODBC Socket	No
Sybase 11	No
Sybase 12	Yes
SQL Server 7	Yes
SQL Server 2000	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 191 for instructions on creating JDBC data sources.

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). Examples of EISs include mainframe transaction processing, database systems, and legacy applications not written in the Java programming language. JCA allows you to integrate EISs with application servers and enterprise applications.

JCA defines a standard set of system-level contracts between an application server and EISs to ensure compatibility between them. The resource adapter implements the EIS portion of these system-level contracts.

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.

JCA 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 3.0 specification describes the relationship of JDBC to the SPI specified in JCA.

The JDBC driver supports appropriate JDBC functionality through the JCA SPI by providing resource adapters. The DataDirect 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

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. See 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:/ConnectionFactories");
            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:/ConnectionFactories");
            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 201 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 Properties

Property	Description
allowPrefetch	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. The initial default value is 0. 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. When set to 0, the prefetch feature is disabled. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Properties *(cont.)*

Property	Description
ApplicationName	Identifies the application that is establishing the connections. When the application does not provide a value, the initial default value is SequeLink for JDBC Application. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
blockFetchForUpdate	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. The initial default value is 1. When set to 0, the appropriate row is locked. When set to 1, the appropriate row is not locked. IMPORTANT: Specifying 0 will degrade performance for SELECT FOR UPDATE statements because rows will be fetched one at a time. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Properties *(cont.)*

Property	Description
certificateChecker	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 com.ddtek.sequelink.cert.CertificateCheckerInterface interface. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties For more information about certificate checker classes, refer to the *SequeLink Administrator's Guide*.
cipherSuites	The Secure Socket Layer (SSL) cipher suites with which the JDBC Client can use to connect. This property is required when networkProtocol=ssl. For a list of supported cipher suites, refer to the *SequeLink Administrator's Guide*. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
databaseName	The name of the data store to which you want to connect. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source

Table 5-3. JDBC Properties *(cont.)*

Property	Description
DBPassword	The data store password, which may be required depending on the server configuration. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
DBUser	The data store user name, which may be required depending on the server configuration. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
description	A description of the connection or data source. This property can be specified using: ■ JDBC data source
HPassword	The host password, which may be required depending on the server configuration. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
HUser	The host user name, which may be required depending on the server configuration. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties

Table 5-3. JDBC Properties *(cont.)*

Property	Description
insensitiveResultSetBuffer	The memory caching scheme for scroll-insensitive cursors. The initial default value is 2058 KB. When set to 0, the driver uses a memory caching mechanism that does not use disk overflow. When set to a value greater than 0, data overflows to disk when the size of cached data exceeds the specified amount, specified in kilobytes. When set to a value less than 0, the data provider provides better performance. However, memory use may be affected. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
MSSMapLongtoDecimal	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 *SequeLink Administrator's Guide*. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
networkProtocol	networkProtocol={socket | ssl}. Specifies the protocol to be used. The initial default value is socket. When set to socket, SSL encryption is not used. When set to ssl, SSL encryption is used. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties

Table 5-3. JDBC Properties *(cont.)*

Property	Description
newPassword	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: `[DataDirect][SequeLink JDBC driver] [SequeLink Server] The user password was changed successfully` This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties For more information about the SequeLink password change mechanism, refer to the *SequeLink Administrator's Guide*.
ORANumber0IsNumeric	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 *SequeLink Administrator's Guide*. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
password	The host or data store password, which may be required depending on the server configuration. This property can be specified using: ■ getConnection ■ JDBC data source ■ URL ■ java.util.properties

Table 5-3. JDBC Properties *(cont.)*

Property	Description
portNumber	The TCP/IP port on which the SequeLink service is listening. This property can be specified using: ■ JDBC data source ■ URL
serverDataSource	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. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties
serverName	The TCP/IP address of the SequeLink server in dotted format or host name format. This property can be specified using: ■ JDBC data source ■ URL
SLKStaticCursorLongColBuffLen	The amount of data (in KB) that is buffered for SQL_LONGVARCHAR and SQL_LONGVARBINARY columns with an insensitive result set. The initial default value is 4. This property can be specified using: ■ JDBC data source ■ URL ■ java.util.properties ■ server data source
spyAttributes	A property that enables Spy when making connections with a JDBC data source. This property only can be specified using a JDBC data source.

Table 5-3. JDBC Properties (cont.)

Property	Description
user	The host or data store user name, which may be required depending on the server configuration. This property can be specified using: ■ getConnection ■ JDBC data source ■ URL ■ java.util.properties

Testing JDBC Connections

See Chapter 6 "Using DataDirect Test" on page 211 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";
permission java.security.SecurityPermission "putProviderProperty.SLJCE";
permission java.security.SecurityPermission "insertProvider.SLJCE";
};
```

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/jdbctest/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.

  Under Windows 9*x* and Me, double-clicking testforjdbc.bat opens a DOS window and displays the error "Out of environment space." To prevent this, use the following procedure:

  a After installing JDBC driver, locate the testforjdbc.bat file in the testforjdbc directory beneath the installation directory.

  b Right-click testforjdbc.bat and select Properties. After the properties display, select the Memory tab.

  c On the Memory tab, locate the Initial environment setting. From this drop-down list, select 1024. Then, select the Protected check box. Click **OK**. A DataDirect Test shortcut is created in the same directory with testforjdbc.bat.

  d Double-click either testforjdbc.bat or the DataDirect Test shortcut. DataDirect Test will open normally without producing the error.

  NOTE: Do not delete the DataDirect Test shortcut; the 1024 environment setting will be lost if the shortcut is deleted.

- **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 through 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 the Chapter 8 “Developing JDBC Applications” on page 271 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 through 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 271 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.

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 221, 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.

The 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 require a Java 2 Platform (JDK 1.2)-compatible Java Virtual Machine.

1 From the Connection window menu, select **Connection** / **Create JDBC 2.0 Statement**. DataDirect Test prompts 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.

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 (JDK 1.2 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**.

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 221, you see that the previously inserted records are returned.

.

Returning ParameterMetaData

NOTE: Returning ParameterMetaData is a JDBC 3.0 feature and requires a JDK 1.4 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 JDK 1.4 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.

Click **Submit**.

5 Select **Connection / Set Savepoint**. In the Set Savepoints window, specify a savepoint name.

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.

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.

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 is a JDBC 3.0 feature and requires a JDK 1.4 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.

6 Deselect Auto Traverse. This disables automatic traversal to the next row.

7 Click **Get Cell**.

8 Values are returned in the Get Cell Value field.

9 Change the Data Type to Clob.

10 Click **Get Cell**. The Clob Data window appears.

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 Spy

Spy is a development software component for tracking JDBC calls. Spy passes calls issued by an application to an underlying JDBC driver and logs detailed information about those calls. It provides the following advantages:

- Spy can only be used with the SequeLink *for* JDBC Driver and the DataDirect Technologies Connect *for* JDBC driver.
- 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, using the SequeLinkDataSource object.

Loading the 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 sql.drivers using the Java -D option. The sql.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 sql.drivers property can be set like other Java properties, using the -D option. For example:

  ```
  java -Dsql.drivers=com.ddtek.jdbcspy.SpyDriver
  ```

- *Method 2*: Set the Java property sql.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 ("sql.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");
  ```

Spy URL Syntax and Spy Attributes

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:

log=System.out	Redirects logging to the Java output standard.
log=(file)*filename*	Redirects logging to the file specified by *filename*. By default, Spy will use the stream specified in DriverManager.setLogStream().
load=*classname*	Loads the driver specified by *classname*. The default value is com.ddtek.sequelink.jdbc. SequeLinkDriver.
linelimit=*numberofchars*	The maximum number of characters, specified by *numberofchars*, that Spy will log on one line. The default is 0 (no maximum limit).
logLobs={yes | no}	Specifies whether Spy logs activity on Blob / Clob. The default is no.

logIS={yes | no | nosingleread}	Specifies whether Spy logs activity on InputStreams. nosingleread=turns on logging for InputStreams (but not InputStream.read() messages), allowing logging on InputStreams without generating large log files full of single-byte read messages. The default is no.
logTName={yes | no}	Specifies whether Spy logs the name of the current thread. The default is no.
timestamp={yes | no}	Specifies whether a timestamp should be included on each line of the DataDirect Spy log.

Using 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.sequelink.SequeLinkDataSource class supports the SpyAttributes connection attribute, which specifies a semi-colon-separated list of Spy attributes as described in "Spy URL Syntax and Spy Attributes" on page 265. See "JDBC Connection URL Examples:" on page 189 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");
...
```

Spy URL Example

jdbc:spy:{*jdbc:sequelink://...*};log=System.out;linelimit=80

Using this example, Spy would:

1. Load the JDBC driver.
2. Log all JDBC activity to the standard output file.
3. Log a maximum of 80 characters for each line.

Spy Log Example

NOTE: Numbers in bold superscript are note indicators. See the notes following the example for the referenced text.

```
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:datadirect:sequelink://QANT:4003;databaseName=
Oracle;
OSUser=qauser;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>> OK[6]

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]
```

SequeLink Developer's Reference

```
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>> OK[19]

spy>> Connection[1].close()
spy>> OK[20]
```

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.

8 Developing JDBC Applications

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

- "JDBC 3.0 Support" on page 272
- "SQL Escape Sequences" on page 273
- "Data Types and Isolation Levels" on page 273
- "Threading" on page 274
- "Using Scrollable Cursors" on page 276
- "Specifying Application IDs" on page 279
- "Error Handling" on page 281
- "Fine-Tuning JDBC Application Performance" on page 283

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 Virtual Machines.

Table 8-1 summarizes the JDBC 3.0 and enhanced JDBC 2.0 functionality that the SequeLink *for* JDBC driver supports.

Table 8-1. Supported JDBC Features

	DB2 z/OS V6, 7	DB2 UDB	Informix	Oracle8, 9i	SQL Server 7, 2000	Sybase
Blob interfaces	X	X	X	X		
Clob interfaces	X	X	X	X		
DatabaseMetaData getSchema			X		X	X
setQueryTimeout		Note 1	X	X	X	X
ParameterMetaData	X	X	X		X	X
Savepoints	X	Note 2		X	X	X
Updatable result sets	X	X	X	X	X	X

NOTES:

1 setQueryTimeout is supported on DB2 UDB only on Windows.

2 Only one savepoint can be released.

See "JDBC Support" on page 135 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 (JCA) Resource Adapter" on page 196 for information about using the SequeLink *for* JDBC driver as a JCA resource adapter.

SQL Escape Sequences

See Appendix A "SQL Escape Sequences" on page 357 for information about the SQL escape sequences supported by the SequeLink *for* JDBC driver.

Data Types and Isolation Levels

The data types and isolation levels supported by the SequeLink *for* JDBC driver depend on the data store to which you are connecting. See Appendix B "Data Types and Isolation Levels" on page 377 for database-specific information about data types and isolation levels.

Threading

The SequeLink *for* 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 will return `SQL_SUCCESS`. If the cancel method 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 8-2. Using Cancel in Multithreaded JDBC Applications

Data Store	SQLCancel
DB2 V5, V6, V7 on OS/390	Ignored
DB2 UDB V7.2, V8 on Windows	OK
DB2 UDB V7.2, V8 on UNIX	Ignored
Informix 9, Informix 2000	OK
Microsoft SQL Server 7.0, 2000	OK
Oracle8, Oracle9i on Windows NT, Windows 200*x*, Windows XP	Ignored
Oracle8, Oracle9i on UNIX	OK
Sybase 11, 12	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.

The type of result sets that can be used depend on the data store to which you are connecting. Table 8-3 shows the type of result sets supported for each database.

Table 8-3. Support for Scrollable Cursors (JDBC)

Database	Scroll-Insensitive	Scroll-Sensitive
DB2 on z/OS	X	
DB2 UDB	X	
Informix	X	X
Microsoft SQL Server (see note)	X	X
ODBC Socket	X	
Oracle	X	X
Sybase (see note)	X	X

NOTE: To use scroll-sensitive cursors with Microsoft SQL Server and 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 SequeLink *for* 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 SequeLink *for* JDBC driver supports updatable result sets.

NOTE: When the SequeLink *for* 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 SequeLink *for* 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*.

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 DataDirect Connect *for* JDBC drivers automatically perform 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*.

Error Handling

The SequeLink *for* 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

SequeLink *for* JDBC Driver Errors

An error generated by the SequeLink *for* 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 SequeLink *for* 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 SequeLink *for* 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 SequeLink *for* 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 = WSrsmd.getColumnCount();
...
int ctype = WSrsmd.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.

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 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 update*XXX* 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 getBestRowIndentifier()

Use getBestRowIndentifier() 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 getBestRowIndentifier().

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 getBestRowIndentifier() 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 getBestRowIndentifier().

Consider the previous example again:

```
...
ResultSet WSrowid = getBestRowIndentifier()
   (.... "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 getBestRowIndentifier() 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 301 provides information about using .NET applications with the SequeLink *for* .NET Client.
- Chapter 10 “Developing .NET Applications” on page 319 provides information about developing .NET applications for SequeLink environments.

9 Using the .NET Client

This chapter provides information on using the SequeLink *for* .NET data provider (the .NET data provider).

About the .NET Data Provider

The SequeLink *for* .NET data provider 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 data provider is optimized for the .NET environment, allowing you to incorporate .NET technology and extend the functionality and performance of your existing system.

The machine must have the .NET Framework version 1.1 or higher installed.

The .NET data provider is compliant with the .NET 1.0 Framework Software Development Kit (SDK). The data provider also supports tracing.

Specifying Connection Properties

NOTE: The .NET data provider attaches a data provider-specific prefix to the public .NET objects. For example, the Connection object is SequeLinkConnection.

You connect to a database through the SequeLinkConnection object, using the method SequeLinkConnection.Open(). The following example uses the SequeLinkConnection object to connect to the SequeLink server while passing the user name and password:

```
SequeLink Connection slcon=new SequeLinkConnection
                ("Host=server1;Database=hal;User ID=john;
                  password=beach");
slcon.Open();
```

Using Connection Pooling

The data provider automatically enables connection pooling for your .NET client application. 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.

You can control connection pooling behavior by using connection string attributes (see "Specifying Connection Attributes" on page 305).

Creating a Connection Pool

Each connection pool is associated with a specific connection string. When a connection is closed, a connection pool that associates itself with the connection's distinctive connection string is created. If a new connection is opened and the connection string does not 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 example, 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.

```
SequeLinkConnection conn1 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                    User ID=john;password=beach";
                    conn.Open();
// Pool A is created.

SequeLinkConnection conn2 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                     User ID=mary;password=jtb25cnc";
                    conn.Open();
// Pool B is created because the connection strings differ.

SequeLinkConnection conn3 = new SequeLinkConnection();
conn.ConnectionString = "Host=Accounting;
                    User ID=john;password=beach";
                    conn.Open();
```

Adding Connections

A connection pool is created for each unique connection string. Connections are added to the pool as needed, up to the maximum pool size.

When an application requests a SequeLinkConnection object (the data provider attaches a provider-specific prefix to the public .NET objects), it is obtained from the pool if an eligible connection is available. To be eligible, the connection must currently be unused, and have a valid link to the server.

IMPORTANT: Close the connection when you are finished using it, using either the Close() or Dispose() methods of the SequeLinkConnection object. If you do not explicitly close the connection, it will not be added or returned to the pool.

Removing Connections

A connection is removed from a connection pool when it either exceeds its lifetime as determined by the Connection Lifetime connection string attribute, or when a new connection is initiated by the application (SequeLinkConnection.Open() is called) that has a matching connection string.

When connections are removed from the connection pool, the pool manager checks to see if the connection has been closed at the server. If a connection is no longer valid, the connection is discarded by the pool manager and another connection from the pool is returned if available and valid.

Specifying Connection Attributes

You can modify a connection by specifying connection attributes. See “.NET Public Objects/Interfaces Supported” on page 326 for information about specifying attributes through the Client’s SequeLinkConnection object.

Table 9-1 lists the connection attributes supported by the .NET Client, and describes each attribute. The connection strings have the form:

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

For example, in the following example, the connection string can be used to connect to the DB2 host hal.

```
"Host=hal;Port=19998;User Id=test01;Password=test01;Database=test;"
```

NOTE: All connection string attribute names are case-insensitive. For example, Password is the same as password. However, the values of some attributes, such as User ID and Password, may be case-sensitive.

Table 9-1 gives the names for each attribute, 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. Connection String Attributes for the .NET Client

Attribute	Description
Connection Lifetime	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 attribute can cause some connections to ignore this value. Valid values are 0 to 65335. The initial default value is 0. The lifetime is never limited by time.
Connection Reset	Connection Reset={True | False}. Specifies whether a connection that is removed from the connection pool for reuse will have its state reset to the state of the new Connection object. For example, during its original use, the connection could have had its database changed by the application. Resetting the state impacts performance of the reused connections because the new connection must issue additional commands to the server. The initial default value is False.
Database	Specifies the name of the database to which you want to connect.
Enlist	Enlist={True | False}. Specifies whether the Client automatically attempts to enlist the connection in creating the thread's current transaction context. This attribute is enabled only when the .NET Client is installed with a Server license and the optional MS DTC Support components were selected. The initial default value is False.
Host	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 `SequeLinkAppServer`. Or, you can specify an IP address such as `122.23.15.12`. The initial default value is localhost.
License Path	Specifies the fully-qualified path to the DDTek.lic license file. If you do not provide this option, the .NET Client looks for the license file in the application's current directory. If the license file is not found, the .NET Client checks for keys placed in the registry during the installation process; then, the .NET Client looks for the license key in the installation directory. If the license key is still not found, the .NET Client fails to connect. The initial default value is an empty string.

Table 9-1. Connection String Attributes for the .NET Client *(cont.)*

Attribute	Description
Max Number Of Pools	Specifies the maximum number of connection pools that can be in use at a time during the life of the process. Valid values are from 1 to 65335. The initial default value is 100.
Max Pool Size	Specifies the maximum number of connections within a single pool at any time. Valid values are from 1 to 65335. The initial default value is 100.
Min Pool Size	Specifies the minimum number of connections left open in a connection pool. This option allows you to keep a minimum number of database connections open even though some connections have exceeded their Connection Lifetime value. A connection pool may contain fewer connections than the specified value until the value is reached. Valid values are 0 to 65335. The initial default value is 0.
Password	Specifies the host or data store password, which may be required depending on the server configuration.
Pooling	Pooling={True | False}. Specifies whether connections are pooled. When set to True, connection pooling is enabled. The initial default value is True.
Port	Specifies the TCP/IP port on which the SequeLink Server is listening. The initial default value is 19996.
ServerDatasource	Specifies 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 .NET Objects

The .NET data provider supports the .NET public objects. See ".NET Public Objects/Interfaces Supported" on page 326 for more information.

The SequeLink *for* .NET data provider 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 data provider supports input parameter arrays for Insert and Update statements. The data provider supports output arrays for stored procedures.

Stored Procedures

NOTE: The .NET data provider 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 data provider attaches the provider-specific prefix "SequeLink" to the names of the public objects, for example, SequeLinkConnection or SequeLinkCommand.

The .NET data provider supports both local and distributed transactions.

Using Local Transactions

The .NET data provider 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 data provider 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 data provider 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 DataDirect Connect *for* .NET data providers support distributed transactions only when the following conditions are met:

- The Server version of DataDirect Connect *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.

Tracing Method Calls

The .NET data provider can trace all its method calls and the returns from those method calls to a user-specified trace file.

Set the following environment variables to enable and control tracing:

DDTek_Enable_Trace	When set to 1 or higher, enables tracing. The default is 0. Tracing is disabled.
DDTek_Trace_File	When set to 1 or higher, specifies the path and name of the trace file. The default is \DDTekTrace.txt.

DDTek_Recreate_Trace	When set to 1 or higher, the Trace file is recreated each time the application is run. The default is 0.

The .NET 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.

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, and *Value* is the hash code of an object appropriate to the individual function calls.

NOTE: For installation instructions, refer to the *SequeLink Installation Guide*.

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 database using the SequeLink .NET Client from an application developed in Visual Studio.NET, using C#. If you are connecting using a different .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 SequeLink .NET Client in the component list.

3 Click **OK**. The Solution Explorer now includes the SequeLink .NET Client.

4 If you are using C#, make sure that the SequeLink namespace appears at the beginning of your application, as shown in the following code fragment:

```
// Access SequeLink Server
using System.Data;
using DDTek.SequeLink;
```

5 Add the connection information for your server and exception handling code, as shown in the following C# code fragment. (See "Specifying Connection Attributes" on page 305 for the connection attributes.)

```
try
{
   DBConn = new SequeLinkConnection("Host=sydney;
                  Port=19998;User ID=test01;
                   Password=test01;Database=test");
   DBConn.Open();
}
```

```
catch (Exception ex)
{
   MessageBox.Show(ex.Message);
}
```

10 Developing .NET Applications

This chapter provides information about developing .NET applications for the SequeLink environment including:

- "Namespace" on page 319
- "Data Types" on page 320
- "Threading" on page 325
- "Event Handling" on page 325
- ".NET Public Objects/Interfaces Supported" on page 326
- ".NET Data Provider Objects" on page 333
- "Setting .NET Security Permissions" on page 345
- "Error Handling" on page 345
- "Designing .NET Applications for Performance" on page 347

Developers of data consumer applications must be familiar with the Microsoft .NET specification and object-oriented programming techniques.

Namespace

The namespace for the SequeLink 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

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( )
CLOB	Clob	String	GetString( )
			GetChars( )
DATE	Date	DateTime	GetDateTime( )

Table 10-1. Mapping of SequeLink Data Types (cont.)

SequeLink Service Data Type	SequeLinkDbType	.NET Framework Type	.NET Framework Typed Accessor
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
Boolean	Bit
Byte	Byte
Currency	Decimal
Date	Date

Table 10-2. Mapping of the System.Data.DbTypes to SequeLinkDbTypes *(cont.)*

System.Data.DbType	SequeLinkDbType
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.

Table 10-3. Mapping .NET Framework Data Types to SequeLinkDbTypes

.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 data types and isolation levels supported by the .NET provider depend on the data store to which you are connecting. See Appendix B "Data Types and Isolation Levels" on page 377 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 341 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 objects, interfaces, properties, and methods, as shown in Table 10-1. The .NET data provider attaches the provider-specific prefix "SequeLink" to the names of the public objects, for example, SequeLinkConnection or SequeLinkCommand.

Table 10-4. Supported .NET Framework Objects/Interfaces for the .NET Data Provider

Objects/Interfaces	Properties	Methods
Command	■ CommandText ■ CommandTimeout ■ CommandType ■ Connection ■ Parameters ■ Transaction ■ UpdatedRowSource	■ Cancel ■ CreateParameter ■ Dispose ■ ExecuteNonQuery ■ ExecuteReader ■ ExecuteScalar ■ Prepare
CommandBuilder	■ DataAdapter ■ QuotePrefix ■ QuoteSuffix	■ DeriveParameters ■ GetDeleteCommand ■ GetInsertCommand ■ GetUpdateCommand ■ RefreshSchema
Connection	■ ConnectionString ■ ConnectionTimeout ■ Database ■ State	■ BeginTransaction ■ ChangeDatabase ■ Close ■ CreateCommand ■ Dispose ■ Open
DataAdapter	■ DeleteCommand ■ InsertCommand ■ MissingMappingAction ■ MissingSchemaAction ■ SelectCommand ■ TableMappings ■ UpdateCommand	■ Fill ■ FillSchema ■ GetFillParameters ■ Update

Table 10-4. Supported .NET Framework Objects/Interfaces for the .NET Data Provider *(cont.)*

Objects/Interfaces	Properties	Methods	
DataReader	■ Depth ■ IsClosed ■ RecordsAffected	■ Close ■ Dispose ■ GetBoolean ■ GetBytes ■ GetChar ■ GetChars ■ GetData ■ GetDateTime ■ GetDataTypeName ■ GetDecimal ■ GetDouble ■ GetFieldType ■ GetSchemaTable ■ GetTimeSpan	■ GetFloat ■ GetGUID ■ GetInt16 ■ GetInt32 ■ GetInt64 ■ GetName ■ GetOrdinal ■ GetString ■ GetValue ■ GetValues ■ IsDBNull ■ NextResult ■ Read
Error	■ Class ■ Errors ■ LineNumber ■ Message ■ Number ■ Procedure	■ N/A	
ErrorCollection	■ Count ■ Item	■ N/A	
Exception	■ HelpLink ■ InnerException ■ Message ■ Source ■ StackTrace ■ TargetSite	■ GetBaseException ■ GetObjectData ■ ToString	
InfoEventMessageArgs	■ Errors ■ Message	■ N/A	

Table 10-4. Supported .NET Framework Objects/Interfaces for the .NET Data Provider *(cont.)*

Objects/Interfaces	Properties	Methods	
Parameter	■ DbType ■ Direction ■ IsNullable ■ ParameterName ■ Precision ■ Scale ■ Size ■ SourceColumn ■ SourceVersion ■ Value	■ N/A	
ParameterCollection	■ Count ■ Item	■ Add ■ Clear ■ Contains ■ CopyTo ■ GetEnumerator	■ GetEnumerator ■ IndexOf ■ RemoveAt
Transaction	■ Connection ■ IsolationLevel	■ Commit ■ Dispose ■ Rollback ■ Save	

For more information on public objects, refer to the Microsoft .NET Framework documentation.

GetSchemaTable Method

The result set produced for the SequeLinkDataReader.GetSchemaTable() returns the columns described in Table 10-5, in the order defined.

Table 10-5. Columns Returned by GetSchemaTable on DataReader

Column	Description
ColumnName	Specifies the name of the column, which might not be unique. If the name cannot be determined, a null value is returned. This name reflects the most recent renaming of the column in the current view or command text.
ColumnOrdinal	Specifies the ordinal of the column, which cannot be NULL. The bookmark column of the row, if any, is 0. Other columns are numbered starting with 1.
ColumnSize	Specifies the maximum possible length of a value in the column. For columns that use a fixed-length data type, this is the size of the data type.
NumericPrecision	Specifies the precision of the column, which depends on how the column is defined in ProviderType. If ProviderType is a numeric data type, this is the maximum precision of the column. If ProviderType is not a numeric data type, this is a null value.
NumericScale	Specifies the number of digits to the right of the decimal point If ProviderType is DBTYPE_DECIMAL or DBTYPE_NUMERIC. Otherwise, this is a null value. The value depends on how the column is defined in ProviderType.
IsUnique	IsUnique is guaranteed to be true if the column constitutes a key by itself or if there is a constraint of type UNIQUE that applies only to this column. The value is True if no two rows in the base table (the table returned in BaseTableName) can have the same value in this column. The initial default value for this column is False. The column can contain duplicate values in the base table.

Table 10-5. Columns Returned by GetSchemaTable on DataReader *(cont.)*

Column	Description
IsKey	The set of columns with IsKey set to true must uniquely identify a row in the rowset. This set of columns can be, but is not required to be, a minimal set of columns. This set of columns may be generated from a base table primary key, a unique constraint or a unique index. The value is True if the column is one of a set of columns in the rowset that, taken together, uniquely identify the row. The value is False if the column is not required to uniquely identify the row.
BaseServerName	Specifies the name of the server that contains the column. The value is null if the base server name cannot be determined. The initial default value of this column is a null value.
BaseCatalogName	Specifies the name of the catalog in the data store that contains the column. A null value if the base catalog name cannot be determined. The initial default value is null.
BaseColumnName	Specifies the name of the column in the data store. This might be different than the column name returned in the ColumnName column if an alias was used. A null value is used if the base column name cannot be determined or if the rowset column is derived from, but is not identical to, a column in the database. The initial default value of this column is a null value.
BaseSchemaName	Specifies the name of the schema in the database that contains the column. The value is null if the base schema name cannot be determined. The initial default value of this column is null.
BaseTableName	Specifies the name of the table or view in the data store that contains the column. A null value is used if the base table name cannot be determined. The initial default value of this column is a null value.
DataType	Maps to the .Net Framework type of the column.

Table 10-5. Columns Returned by GetSchemaTable on DataReader *(cont.)*

Column	Description
AllowDBNull	Set if the consumer can set the column to a null value, or if the data provider cannot determine whether or not the consumer can set the column to a null value. Otherwise, no value is set. A column can contain null values, even if it cannot be set to a null value.
ProviderType	Specifies the indicator of the column's data type. This column cannot contain a null value. If the data type of the column varies from row to row, this must be Object. The value is an int that refers to the SequeLinkDbType, as shown below: ■ -7 Bit ■ -6 Byte ■ -5 BigInt ■ -4 LVarBinary ■ -3 VarBinary ■ -2 Binary ■ -1 LVarChar ■ 1 Char ■ 3 Decimal ■ 4 Int ■ 5 SmallInt ■ 7 Single ■ 8 Double ■ 12 VarChar ■ 91 Date ■ 92 Time ■ 93 Timestamp ■ 2004 Blob ■ 2005 Clob
IsAliased	Specifies whether a column is aliased. The value is True if the column is aliased; otherwise, the value is False.
IsExpression	Specifies whether a column is an expression. The value is True if the column is an expression; otherwise, the value is False.

Table 10-5. Columns Returned by GetSchemaTable on DataReader (cont.)

Column	Description
IsIdentity	Specifies whether a column is an identity column. The value is True if the column is an identity column; otherwise, the value is False.
IsAutoIncrement	Specifies whether the column assigns values to new rows in fixed increments. VARIANT_TRUE: The column assigns values to new rows in fixed increments. VARIANT_FALSE: The column does not assign values to new rows in fixed increments. The initial default value is VARIANT_FALSE.
IsRowVersion	Set if the column is automatically updated each time any value in the row is updated by any transaction.
IsHidden	Specifies whether a column is hidden. The initial default value is True if the column is hidden; otherwise, the value is False.
IsLong	Set if the column contains a Binary Long Object (BLOB) that contains very long data. The setting of this flag corresponds to the value of the IS_LONG column in the PROVIDER_TYPES rowset for the data type. The definition of very long data is provider-specific.
IsReadOnly	Determines whether a column can be changed. The value is false if the column can be modified; otherwise, the value is true.

.NET Data Provider Objects

The .NET data provider supports the public .NET objects, properties, and methods. For more information, refer to the Microsoft .NET programming documentation.

See Appendix E ".NET Code Examples" on page 475 for sample code that uses these objects.

The .NET data provider attaches the provider-specific prefix SequeLink to the public .NET objects, for example, SequeLinkCommand.

In addition to the public objects, the SequeLink data provider also supports the provider-specific SequeLinkInfoMessageEventArgs object.

SequeLinkCommand Object

In addition to the public properties of the Command object, the SequeLinkCommand object supports the property described in Table 10-6.

Table 10-6. Properties of the SequeLinkCommand Object

Property	Description
ArrayBindCount	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. The initial default value is 0. The application will not use parameter array binding.

Table 10-6. Properties of the SequeLinkCommand Object *(cont.)*

Property	Description
ArrayBindStatus	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 OracleRowStatus. 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 OracleRowStatus enumeration has the following possible values: ■ 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	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. Valid values are 0 to 2147483647. When set to 0, the data provider does not limit the number of rows returned. The initial default value is 0.

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.

Table 10-7. Properties of the SequeLinkCommandBuilder Object

Property	Description
AllowUpdateWithout Key	Allows the use of the SequeLinkCommandBuilder on a table that has no primary key or unique index. When set to True, a DataAdapter.Update can succeed when a single SQL statement updates multiple rows in the underlying table. Normally, each Update or Delete statement executed on a DataAdapter.Update changes only one row. The initial default value is False.

SequeLinkConnection Object

In addition to the public properties of the Connection object, the SequeLinkConnection object supports the properties described in Table 10-8. Some properties return the values specified for the corresponding connection string attribute (refer to the *SequeLink Administrator's Guide* for information on the connection string attributes).

Table 10-8. 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 *SequeLink Administrator's Guide*.
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.

You can use the InfoMessage event of the SequeLinkConnection object to retrieve warnings and informational messages from the database. If the database returns an error, an exception is thrown. Clients that want to process warnings and informational messages sent by the database server should create a

SequeLinkInfoMessageEventHandler delegate to register for this event.

The InfoMessage event receives an argument of type SequeLinkInfoMessageEventArgs containing data relevant to this event.

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.

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.

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.

In addition to the public properties of the DataReader object, the SequeLinkDataReader object supports the properties described in Table 10-9.

Table 10-9. Properties of the SequeLinkDataReader Object

Property	Description
FieldCount	Gets the number of columns in the current row
Item	Overloaded. Gets the value of a column in its native format.

SequeLinkError Object

The SequeLinkError object collects information relevant to errors and warnings generated by the SequeLink server.

In addition to the public properties of the Error object, the SequeLinkError object supports the properties described in Table 10-10.

Table 10-10. Properties of the SequeLinkError Object

Property	Description
Class	Gets the severity level of the error returned from the SequeLink server.
Message	Returns the error message text returned from the SequeLink server.
Number	Returns the error number returned from the SequeLink server.

Table 10-10. Properties of the SequeLinkError Object (cont.)

Property	Description
Source	Gets the name of the provider that generated the error.
SQLState	Gets a numeric error code from the SequeLink server. The error code represents an error, warning, or "no data found" message.

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.

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.

Table 10-11. Properties of the SequeLinkException Object

Properties	Description
Errors	Specifies a SequeLinkErrorCollection of one or more SequeLinkError objects.
Number	Specifies the error number returned from the SequeLink server.
SQLState	Gets a numeric error code from the SequeLink server. The error code represents an error, warning, or "no data found" message.

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 345 for an example of using InfoMessage delegates to retrieve warning information.

Table 10-12 provides the provider-specific properties that are supported for the SequeLinkInfoMessageEventArgs object.

Table 10-12. Properties of the SequeLinkInfoMessageEventArgs Object

Property	Description
Errors	Specifies a SequeLinkErrorCollection that contains a collection of warnings sent from the SequeLink server.
Message	Returns the text of the last message returned from the SequeLink server. The application should check the Count property of the SequeLinkErrorCollection returned in the Errors property of this object to determine whether multiple warnings exist.
Source	Gets the name of the object that generated the error.

SequeLinkParameter Object

The SequeLinkParameter object represents a parameter to a SequeLinkCommand.

Table 10-13. Properties of the SequeLinkParameter Object

Property	Description
SequeLinkDbType	Specifies the data types of the SequeLink server. See "Data Types" on page 320 for more information about SequeLink data types.

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 Attributes" on page 305).

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 10-14 provides the properties for the SequeLinkTrace object.

Table 10-14. Properties of the SequeLinkTrace Object

Property	Description
EnableTrace	When set to 1 or higher, enables tracing. The initial default value is 0. Tracing is disabled.

Table 10-14. Properties of the SequeLinkTrace Object

Property	Description
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 value is \SequeLinkTrace.txt. If the file does not exist, the data provider creates it.
RecreateTrace	When set to 1, recreates the trace file each time the application restarts. The initial default value is 0; the trace file is appended.

SequeLinkTransaction Object

The SequeLinkTransaction object implement the IDBTransaction interface as specified in the .NET Framework.

Table 10-15 provides the methods used by the SequeLinkTransaction object to support savepoints.

Table 10-15. 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 *for* .NET Client:

- .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 339 for more information.

See "Retrieving Warning Information" on page 490 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 340 for more information.

SequeLink Client 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.

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.

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.

Understanding the Architecture

ADO.NET uses .NET data providers to access data. All .NET runtime data access is done through .NET data providers.

ADO.NET development wizards use OLE DB to populate the grids of the wizards. Although OLE DB is accessed for these wizards at design time, the runtime components generated use .NET data providers for accessing data. Optimizing data access via OLE DB in a .NET application is not a useful exercise. Instead, optimize the .NET data provider as the component that will do all of the runtime work.

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 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.

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.

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 any data being retrieved to some manageable limit by calling set max rows or set max field size, or some other database-specific command to limit row size or field size. Another method of reducing the size of the data being retrieved is to decrease the column size. If the data provider 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.

Using CommandBuilder Objects

It is tempting to use CommandBuilder objects because they generate SQL statements. However, this shortcut can have a negative effect on performance. Because of concurrency restrictions, the Command Builder does not generate efficient SQL statements.

For example, the following SQL statement was created by the Command Builder:

```
CommandText: UPDATE TEST01.EMP SET EMPNO = ?, ENAME =
?, JOB = ?, MGR = ?, HIREDATE = ?, SAL = ?, COMM =
?, DEPT = ?
WHERE
  ( (EMPNO = ?) AND ((ENAME IS NULL AND ? IS NULL)
  OR (ENAME = ?)) AND ((JOB IS NULL AND ? IS NULL)
  OR (JOB = ?)) AND ((MGR IS NULL AND ? IS NULL)
  OR (MGR = ?)) AND ((HIREDATE IS NULL AND ? IS NULL)
  OR (HIREDATE = ?)) AND ((SAL IS NULL AND ? IS NULL)
  OR (SAL = ?)) AND ((COMM IS NULL AND ? IS NULL)
  OR (COMM = ?)) AND ((DEPT IS NULL AND ? IS NULL)
  OR (DEPT = ?)) )
```

The end-user can often write more efficient update and delete statements than those that the Command Builder generates.

Another drawback is also implicit in the design of the CommandBuilder object. The CommandBuilder object is always associated with a DataAdapter object and registers itself as a listener for RowUpdating and RowUpdated events that the DataAdapter object generates. This means that two events must be processed for every row that is updated.

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 .NET Objects and Methods

The guidelines in this section will help you to optimize system performance when selecting and using .NET 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.

When you set the CommandType property of the command to StoredProcedure, do not physically code the literal arguments into the CommandText. For example, do not use literal arguments such as:

```
{call foo (3567, 'John', 987.32)}
```

.NET data providers can call stored procedures on the database server by executing the procedure as any other SQL query. 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.

In the following example, the application programmer might assume that the only argument to getCustName is the integer 12345:

```
{call getCustName (12345)}
```

However, SQL is always sent to the database server as a character string. When the database server parses the SQL query and isolates the argument value, the result is still a string. The database server must then convert the string ’12345’ into the

integer 12345. Using a parameter marker eliminates the need to convert the string and reduces the amount of processing by the server:

```
{call getCustName (?)}
```

Designing .NET Applications

The guidelines in this section will help you to optimize system performance when designing .NET 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.

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 I/O needed to create a new connection.

Pre-allocate connections. Decide what connection strings you will need to meet your needs. Remember that each unique connection string creates a new connection pool.

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.

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 data provider to be significantly slower than data providers that are completely managed code. Why take that kind of performance hit?

If you use a bridge, your code will be written for this bridge. Later, when a database-specific ADO.NET data provider becomes available; the code must be rewritten — you will have to rewrite object names, schema information, error handling, and parameters. You'll save valuable time and resources by coding to managed data providers instead of to the bridges.

Updating Data

This section provides general guidelines to help you to optimize system performance when updating data in databases.

Using the Disconnected DataSet

Keep result sets small. The full result set must be retrieved from the server before the DataSet is populated. The DataSet must then be converted to XML and processed. The full result set is stored on the client.

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://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconusingadonetproviderstoaccessdata.asp
- *Transaction Control: Building Distributed Transactions with .NET*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdadotnettransact1.asp
- *Writing Serviced Components*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwritingservicedcomponents.asp
- *Creating and Using DataSets*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconcreatingusingdatasets.asp
- *XML and the DataSet*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconxmldataset.asp

Part 5: Reference

This part contains the following appendixes:

- Appendix A "SQL Escape Sequences" on page 357 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 377 lists the data types and isolation levels supported for each data store supported by SequeLink.
- Appendix C "JDBC Support" on page 415 provides information about JDBC compatibility and developing JDBC applications for SequeLink environments.
- Appendix D "JDBC Connection Pool Manager" on page 465 describes how your application can use connection pooling through the DataDirect Connection Pool Manager.
- Appendix E ".NET Code Examples" on page 475 includes code examples of typical database access tasks in ADO.NET.

A SQL Escape Sequences

A number of language features, such as outer joins and scalar function calls, are commonly implemented by DBMSs. The syntax for these features is often DBMS-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.

- LIKE predicate escape characters
- Procedure call escape sequences

NOTE: The .NET provider supports ODBC/JDBC SQL escape sequences, except where otherwise noted.

Table A-1. Language Features

Language Feature	SequeLink *for* ODBC	SequeLink *for* JDBC	SequeLink *for* .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 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.

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.

Example:

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

Table A-2. Scalar Functions Supported on DB2

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
DB2 V5 on OS/390	CONCAT LEFT LENGTH RIGHT SUBSTRING	Not supported	CURDATE CURRENT_DATE CURTIME DAYOFMONTH HOUR MINUTE MONTH NOW QUARTER SECOND YEAR	IFNULL USERNAME

Table A-2. Scalar Functions Supported on DB2 *(cont.)*

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
DB2 V6, V7 on OS/390	CHAR	ABS	CURDATE	DBNAME
	CONCAT	ACOS	CURRENT_DATE	IFNULL
	DIFFERENCE	ASIN	CURTIME	USERNAME
	INSERT	ATAN	DAYOFMONTH	
	LCASE	ATAN2	DAYOFWEEK	
	LEFT	CEILING	DAYOFYEAR	
	LENGTH	COS	HOUR	
	LOCATE	DEGREES	MINUTE	
	LOCATE_2	EXP	MONTH	
	LTRIM	FLOOR	NOW	
	REPEAT	LOG	SECOND	
	REPLACE	LOG10	WEEK	
	RIGHT	MOD	YEAR	
	RTRIM	PI		
	SOUNDEX	POWER		
	SPACE	RADIANS		
	SUBSTRING	RAND		
	UCASE	ROUND		
		SIGN		
		SIN		
		SQRT		
		TAN		
		TRUNCATE		

Table A-2. Scalar Functions Supported on DB2 *(cont.)*

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
DB2 UDB V7.2, 8.1	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_ TIMESTAMP	
	DIFFERENCE	ATAN2	CURTIME	
	INSERT	CEILING	DAYOFMONTH	
	LCASE	COS	DAYOFWEEK	
	LEFT	COT	DAYOFYEAR	
	LENGTH	DEGREES	DAYNAME	
	LOCATE	EXP	EXTRACT	
	LOCATE2	FLOOR	HOUR	
	LTRIM	LOG	MINUTE	
	OCTET_LENGTH	LOG10	MONTH	
	REPEAT	MOD	MONTHNAME	
	REPLACE	PI	NOW	
	RIGHT	POWER	QUARTER	
	RTRIM	RADIANS	SECOND	
	SOUNDEX	RAND	TIMESTAMPADD	
	SPACE	ROUND	TIMESTAMPDIFF	
	SUBSTRING	SIGN	WEEK	
	UCASE	SIN	YEAR	
		SQRT		
		TAN		
		TRUNCATE		

Table A-5. Scalar Functions Supported on Oracle

Data Store	String Functions	Numeric Functions	Timedate Functions	System Functions
Oracle	ASCII	ABS	CURDATE	IFNULL
	BIT_LENGTH	ACOS	CURRENT_DATE	USER*
	CHAR	ASIN	CURRENT_TIME	USERNAME
	CONCAT	ATAN	CURRENT	
	INSERT	ATAN2	TIMESTAMP	
	LCASE	CEILING	DAYOFMONTH	
	LEFT	COS	DAYOFWEEK	
	LENGTH	COT	DAYOFYEAR	
	LOCATE	EXP	DAYNAME	
	LOCATE2	FLOOR	HOUR	
	LTRIM	LOG	MINUTE	
	OCTET_LENGTH	LOG10	MONTH	
	REPEAT	MOD	MONTHNAME	
	REPLACE	PI	NOW	
	RIGHT	POWER	QUARTER	
	RTRIM	ROUND	SECOND	
	SOUNDEX	SIGN	TIMESTAMP_ADD	
	SPACE	SIN	TIMESTAMP_DIFF	
	SUBSTRING	SQRT	WEEK	
	UCASE	TAN	YEAR	
		TRUNCATE		

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[1]	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(*string_exp*)	The ASCII code of the leftmost character of *string_exp* as an integer.
BIT_LENGTH(*string_exp*)	The length, in bits, of the string expression.
CHAR(*code*)	The character with the ASCII code specified by *code*. *code* should be between 0 and 255; otherwise, the return value depends on the data source.
CHAR_LENGTH(*string_exp*)	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(*string_exp*)	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(*string_exp1*, *string_exp2*)	The string resulting from concatenating *string_exp2* and *string_exp1*. The string is system dependent.
DIFFERENCE(*string_exp1*, *string_exp2*)	An integer indicating the difference between the values returned by the SOUNDEX function for *string_exp1* and *string_exp2*.
INSERT(*string_exp1*, *start*, *length*, *string_exp2*)	A string where *length* characters have been deleted from *string_exp1* beginning at *start* and where *string_exp2* has been inserted into *string_exp* beginning at start.
LCASE(*string_exp*)	Uppercase characters in *string_exp* converted to lowercase.
LEFT(*string_exp*,*count*)	The *count* of characters of *string_exp*.
LENGTH(*string_exp*)	The number of characters in *string_exp*, excluding trailing blanks and the string termination character.
LOCATE(*string_exp1*, *string_exp2*[,*start*])	The starting position of the first occurrence of *string_exp1* in *string_exp2*. If *start* is not specified, the search begins with the first character position in *string_exp2*. If *start* is specified, the search begins with the character position indicated by *start*. The first character position in *string_exp2* is indicated by 1. If *string_exp1* is not found, 0 is returned.
LTRIM(*string_exp*)	The characters of *string_exp*, with leading blanks removed.
OCTET_LENGTH(*string_exp*)	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.
POSITION(*character_exp* IN *character_exp*)	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.

Table A-7. Scalar String Functions *(cont.)*

Function	Returns
REPEAT(*string_exp*, *count*)	A string composed of *string_exp* repeated *count* times.
REPLACE(*string_exp1*, *string_exp2*, *string_exp3*)	Replaces all occurrences of *string_exp2* in *string_exp1* with *string_exp3.*
RIGHT(*string_exp*, *count*)	The rightmost *count* of characters in *string_exp*.
RTRIM(*string_exp*)	The characters of *string_exp* with trailing blanks removed.
SOUNDEX(*string_exp*)	A data-source dependent string representing the sound of the words in *string_exp*.
SPACE(*count*)	A string consisting of *count* spaces.
SUBSTRING(*string_exp*, *start*, *length*)	A string derived from *string_exp*, beginning at the character position *start* for *length* characters.
UCASE(*string_exp*)	Lowercase characters in *string_exp* 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(*numeric_exp*)	Absolute value of *numeric_exp*.
ACOS(*float_exp*)	Arccosine of *float_exp* as an angle in radians.
ASIN(*float_exp*)	Arcsine of *float_exp* as an angle in radians.
ATAN(*float_exp*)	Arctangent of *float_exp* as an angle in radians.
ATAN2(*float_exp1*, *float_exp2*)	Arctangent of the x and y coordinates, specified by *float_exp1* and *float_exp2* as an angle in radians.
CEILING(*numeric_exp*)	Smallest integer greater than or equal to *numeric_exp*.
COS(*float_exp*)	Cosine of *float_exp* as an angle in radians.
COT(*float_exp*)	Cotangent of *float_exp* as an angle in radians.
DEGREES(*numeric_exp*)	Number if degrees converted from *numeric_exp* radians.
EXP(*float_exp*)	Exponential value of *float_exp*.
FLOOR(*numeric_exp*)	Largest integer less than or equal to *numeric_exp*.
LOG(*float_exp*)	Natural log of *float_exp*.
LOG10(*float_exp*)	Base 10 log of *float_exp*.
MOD(*integer_exp1*, *integer_exp2*)	Remainder of *integer_exp1* divided by *integer_exp2*.
PI()	Constant value of pi as a floating-point number.
POWER(*numeric_exp*, *integer_exp*)	Value of *numeric_exp* to the power of *integer_exp*.
RADIANS(*numeric_exp*)	Number of radians converted from *numeric_exp* degrees.
RAND([*integer_exp*])	Random floating-point value using *integer_exp* as the optional seed value.
ROUND(*numeric_exp*, *integer_exp*)	*numeric_exp* rounded to *integer_exp* places right of the decimal (left of the decimal if *integer_exp* is negative).

Table A-8. Scalar Numeric Functions *(cont.)*

Function	Returns
SIGN(*numeric_exp*)	Indicator of the sign of *numeric_exp*. If *numeric_exp* < `0`, `-1` is returned. If *numeric_exp* = `0`, `0` is returned. If *numeric_exp* > `0`, `1` is returned.
SIN(*float_exp*)	Sine of *float_exp*, where *float_exp* is an angle in radians.
SQRT(*float_exp*)	Square root of *float_exp*.
TAN(*float_exp*)	Tangent of *float_exp*, where *float_exp* is an angle in radians.
TRUNCATE(*numeric_exp*, *integer_exp*)	*numeric_exp* truncated to *integer_exp* places right of the decimal. (If *integer_exp* 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[(*time-precision*)]	Current local time. The *time-precision* argument determines the seconds precision of the returned value.
CURRENT_TIMESTAMP[(*timestamp-precision*)]	Current local date and local time as a timestamp value. The *timestamp-precision* argument determines the seconds precision of the returned timestamp.
CURTIME()	Current local time as a time value.
DAYNAME(*date_exp*)	Character string containing a data-source-specific name of the day for the day portion of *date_exp*.
DAYOFMONTH(*date_exp*)	Day of the month in *date_exp* as an integer value (1–31).
DAYOFWEEK(*date_exp*)	Day of the week in *date_exp* as an integer value (1–7).
DAYOFYEAR(*date_exp*)	Day of the year in *date_exp* as an integer value (1–366).
EXTRACT(*extract-field FROM extract-source*)	*Extract-field* portion of the *extract-source*. The *extract-source* argument is a datetime or interval expression. The *extract-field* argument can be one of the following keywords: YEAR MONTH DAY HOUR MINUTE SECOND 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 *extract-source* field.
HOUR(*time_exp*)	Hour in *time_exp* as an integer value (0–23).

Table A-9. Scalar Time and Date Functions *(cont.)*

Function	Returns
MINUTE(*time_exp*)	Minute in *time_exp* as an integer value (0–59).
MONTH(*date_exp*)	Month in *date_exp* as an integer value (1–12).
MONTHNAME(*date_exp*)	Character string containing the data source–specific name of the month.
NOW()	Current date and time as a timestamp value.
QUARTER(*date_exp*)	Quarter in *date_exp* as an integer value (`1-4`).
SECOND(*time_exp*)	Second in *time_exp* as an integer value (`0-59`).
TIMESTAMPADD(*interval*, *integer_exp*, *time_exp*)	Timestamp calculated by adding *integer_exp* intervals of type *interval* to *time_exp*. interval can be one of the following values: SQL_TSI_FRAC_SECOND SQL_TSI_SECOND SQL_TSI_MINUTE SQL_TSI_HOUR SQL_TSI_DAY SQL_TSI_WEEK SQL_TSI_MONTH SQL_TSI_QUARTER SQL_TSI_YEAR Fractional seconds are expressed in billionths of a second.
TIMESTAMPDIFF(*interval*, *time_exp1*, *time_exp2*)	Integer number of intervals of type *interval* by which *time_exp2* is greater than *time_exp1*. *interval* has the same value as *TIMESTAMPADD*. Fractional seconds are expressed in billionths of a second.
WEEK(*date_exp*)	Week of the year in *date_exp* as an integer value (1–53).
YEAR(*date_exp*)	Year in *date_exp*. The range is data-source dependent.

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(*exp,value*)	*value*, if *exp* 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 start 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:

```
{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 V5, V6R1, V7R1 on OS/390	Left outer joins Right outer joins Full outer joins Nested outer joins Inner outer joins
DB2 VV7.2, 8.1 on Windows NT, Windows 200x, and UNIX	Left outer joins Right outer joins Full outer joins Nested outer joins Unordered outer joins Inner outer joins
Informix	Left outer joins Unordered outer joins
Microsoft SQL Server	Left outer joins Right outer joins Full outer joins Nested outer joins

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.

NOTE: For DB2 V5 on OS/390, stored procedures cannot be qualified with a schema name.

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. 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.

The relative processing time for data types is shown in the following figure:

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 on OS/390

The following table lists the data types that the ODBC driver supports for DB2 V5, V6, and V7 on OS/390 and z/OS.

Table B-1. Mapping Data Types for DB2 on OS/390 to ODBC Data Types

DB2 V5 Data Type	DB2 V6, V7 Data Type	ODBC Data Type (ANSI)
CHAR(size)	CHAR(size)	SQL_CHAR
		SQL_TIMESTAMP Note 1
CHAR(size) FOR BIT DATA	CHAR(size) FOR BIT DATA	SQL_BINARY
	CLOB	SQL_LONGVARCHAR
DATE	DATE	SQL_DATE
	DBCLOB Note 1	SQL_WLONGVARCHAR
DECIMAL	DECIMAL	SQL_DECIMAL
FLOAT	FLOAT	SQL_FLOAT
	GRAPHIC(size) Note 1	SQL_WCHAR
INTEGER	INTEGER	SQL_INTEGER
LONG VARCHAR	LONGVARCHAR	SQL_LONGVARCHAR
LONG VARCHAR FOR BIT DATA	LONGVARBINARY	SQL_LONGVARBINARY
REAL	REAL	SQL_REAL
	ROWID	SQL_VARBINARY
SMALLINT	SMALLINT	SQL_SMALLINT
TIME	TIME	SQL_TIME
TIMESTAMP	TIMESTAMP	SQL_TIMESTAMP

NOTE 1 The setting of DataSourceFetchTimestampAsString determines whether timestamps are fetched as character strings.

Table B-1. Mapping Data Types for DB2 on OS/390 to ODBC Data Types (cont.)

DB2 V5 Data Type	DB2 V6, V7 Data Type	ODBC Data Type (ANSI)
VARCHAR(size)	VARCHAR(size)	SQL_VARCHAR
VARCHAR(size) FOR BIT DATA	VARCHAR(size) FOR BIT DATA	SQL_VARBINARY

NOTE 1 The setting of DataSourceFetchTimestampAsString determines whether timestamps are fetched as character strings.

The following table lists the data types that the ADO data provider supports for DB2 V5, V6, and V7 on OS/390 and z/OS.

Table B-2. Mapping Data Types for DB2 on OS/390 to ADO Data Types

DB2 V5 Data Type	DB2 V6, 7 Data Type	OLE DB Data Type
Not Supported	BLOB	DBTYPE_BYTES
CHAR(size)	CHAR	DBTYPE_STR
		DBTYPE_DBTIMESTAMP Note 1
CHAR(size) FOR BIT DATA	CHAR(size) FOR BIT DATA	DBTYPE_BYTES
Not Supported	CLOB	DBTYPE_STR
DATE	DATE	DBTYPE_DBDATE
DECIMAL	DECIMAL	DBTYPE_DECIMAL
FLOAT	FLOAT	DBTYPE_R8
INTEGER	INTEGER	DBTYPE_I4
LONG VARCHAR	LONG VARCHAR	DBTYPE_STR
LONG VARCHAR FOR BIT DATA	LONG VARCHAR FOR BIT DATA	DBTYPE_BYTES
REAL	REAL	DBTYPE_R4
Not Supported	ROWID	DBTYPE_BYTES
SMALLINT	SMALLINT	DBTYPE_I2
TIME	TIME	DBTYPE_DBTIME

NOTE 1 The setting of DataSourceFetchTimestampAsString determines whether timestamps are fetched as character strings.

Table B-2. Mapping Data Types for DB2 on OS/390 to ADO Data Types (cont.)

DB2 V5 Data Type	DB2 V6, 7 Data Type	OLE DB Data Type
TIMESTAMP	TIMESTAMP	DBTYPE_DBTIMESTAMP
VARCHAR(size)	VARCHAR(size)	DBTYPE_STR
VARCHAR(size) FOR BIT DATA	VARCHAR(size) FOR BIT DATA	DBTYPE_BYTES

NOTE 1 The setting of DataSourceFetchTimestampAsString determines whether timestamps are fetched as character strings.

The following table lists the data types that the JDBC driver supports for DB2 V5, V6, and V7 on OS/390 and z/OS.

Table B-3. Mapping Data Types for DB2 on OS/390 to JDBC Data Types

DB2 V5 Data Type	DB2 V6, V7 Data Type	JDBC Data Type
	BLOB	LONGVARBINARY Note 1
		BLOB Note 1
CHAR(size)	CHAR(size)	CHAR
CHAR(size) FOR BIT DATA	CHAR(size) FOR BIT DATA	BINARY
	CLOB	LONGVARCHAR Note 1
		CLOB Note 1
DATE	DATE	DATE
DECIMAL	DECIMAL	DECIMAL
FLOAT	FLOAT	FLOAT
INTEGER	INTEGER	INTEGER
LONG VARCHAR	LONG VARCHAR	LONGVARCHAR
LONG VARCHAR FOR BIT DATA	LONG VARCHAR FOR BIT DATA	LONGVARBINARY

1 Depending on the configuration file setting DataSourceJDBCLobAsLongvar, the LOB data types will be handled in JDBC as LOB objects or as Longvar data types. If set to TRUE, LONGVARCHAR data types are reported; if set to FALSE, LOB data types are reported.

2 DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings.

Table B-3. Mapping Data Types for DB2 on OS/390 to JDBC Data Types (cont.)

DB2 V5 Data Type	DB2 V6, V7 Data Type	JDBC Data Type
REAL	REAL	REAL
	ROWID	VARBINARY
SMALLINT	SMALLINT	SMALLINT
TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP
		CHAR Note 2
VARCHAR(size)	VARCHAR(size)	VARCHAR
VARCHAR(size) FOR BIT DATA	VARCHAR(size) FOR BIT DATA	VARBINARY

1 Depending on the configuration file setting DataSourceJDBCLobAsLongvar, the LOB data types will be handled in JDBC as LOB objects or as Longvar data types. If set to TRUE, LONGVARCHAR data types are reported; if set to FALSE, LOB data types are reported.

2 DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings.

The following table maps the data types that the .NET data provider supports for DB2 on OS/390 and z/OS.

Table B-4. Mapping Data Types for DB2 on OS/390 to .NET Data Types

DB2 V5 Data Type	DB2 V6, V7 Data Type	.NET Data Type
	BLOB	byte[]
CHAR	CHAR	String
CHAR(size) FOR BIT DATA	CHAR(size) FOR BIT DATA	byte[]
	CLOB	String
DATE	DATE	DateTime
DECIMAL	DECIMAL	Decimal
FLOAT	FLOAT	Double
INTEGER	INTEGER	Int32
LONG VARCHAR	LONG VARCHAR	String

Table B-4. Mapping Data Types for DB2 on OS/390 to .NET Data Types (cont.)

DB2 V5 Data Type	DB2 V6, V7 Data Type	.NET Data Type
LONG VARCHAR(size) FOR BIT DATA	VARCHAR(size) FOR BIT DATA	byte[]
REAL	REAL	Single
	ROWID	byte[]
SMALLINT	SMALLINT	Int16
TIME	TIME	DateTime
TIMESTAMP	TIMESTAMP	DateTime
		String
VARCHAR	VARCHAR	String
VARCHAR(size) FOR BIT DATA	VARCHAR(size) FOR BIT DATA	byte[]

DB2 UDB V7, V8

The following table lists the data types that the ODBC driver supports for DB2 UDB V7 and V8 for the default service (ServiceCodePage=Default or OS) and the Unicode service (ServiceCodePage=Database).

Table B-5. Mapping DB2 UDB Data Types 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 Note 1	SQL_LONGVARBINARY	SQL_LONGVARBINARY
CHAR(size)	SQL_CHAR	SQL_CHAR Note 3
		SQL_WCHAR
CHAR(size) FOR BIT DATA	SQL_BINARY	SQL_BINARY
CLOB Note 1	SQL_LONGVARCHAR	SQL_WLONGVARCHAR Note 5
DATE	SQL_DATE	SQL_DATE
DBCLOB Note 4	Not supported	SQL_WLONGVARCHAR
DECIMAL	SQL_DECIMAL	SQL_DECIMAL
FLOAT	SQL_DOUBLE	SQL_DOUBLE
	SQL_FLOAT	SQL_FLOAT
GRAPHIC(size) Note 4	Not supported	SQL_WCHAR
INTEGER	SQL_INTEGER	SQL_INTEGER
LONG VARCHAR Note 1	SQL_LONGVARCHAR	SQL_LONGVARCHAR Note 3
LONG VARCHAR FOR BIT DATA Note 1	SQL_LONGVARBINARY	SQL_LONGVARBINARY
LONG VARGRAPHIC Note 4	Not supported	SQL_WLONGVARCHAR
NUMERIC	SQL_NUMERIC	SQL_NUMERIC

NOTES:

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.

Table B-5. Mapping DB2 UDB Data Types to ODBC Data Types *(cont.)*

DB2 SQL Data Type	ODBC Data Types (Non-Unicode Service)	ODBC Data Types (Unicode Service)
REAL	SQL_REAL	SQL_REAL
SMALLINT	SQL_SMALLINT	SQL_SMALLINT
TIME	SQL_TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP	SQL_TIMESTAMP
	SQL_CHAR Note 2	SQL_CHAR Note 2
VARCHAR(size)	SQL_VARCHAR	SQL_VARCHAR Note 3
		SQL_WVARCHAR
VARCHAR(size) FOR BIT DATA	SQL_VARBINARY	SQL_VARBINARY
VARGRAPHIC(size) Note 4	Not supported	SQL_WVARCHAR

NOTES:

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 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 DB2 UDB Data Types to ADO Data Types

DB2 SQL Data Type	OLE DB Data Type Non-Unicode Service
BIGINT	DBTYPE_I8
BLOB Note 1	DBTYPE_BYTES
CHAR	DBTYPE_STR
CHAR() FOR BIT DATA	DBTYPE_BYTES
CLOB Note 1	DBTYPE_STR
DATE	DBTYPE_DBDATE
DECIMAL	DBTYPE_DECIMAL
FLOAT	DBTYPE_R8
INTEGER	DBTYPE_I4
LONG VARCHAR Note 1	DBTYPE_STR
LONG VARCHAR FOR BIT DATA Note 1	DBTYPE_BYTES
NUMERIC	DBTYPE_NUMERIC
REAL	DBTYPE_R4
SMALLINT	DBTYPE_I2
TIME	DBTYPE_DBTIME
TIMESTAMP	DBTYPE_DBTIMESTAMP
	DBTYPE_STR Note 2
VARCHAR(size)	DBTYPE_STR
VARCHAR(size) FOR BIT DATA	DBTYPE_BYTES

NOTES

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 V7 and V8 for the default service (ServiceCodePage=Default or OS) and the Unicode service (ServiceCodePage=Database).

Table B-7. Mapping DB2 UDB Data Types to JDBC Data Types

DB2 SQL Data Type	JDBC Data Type Non-Unicode Service	JDBC Data Type Unicode Service
BIGINT	BIGINT	BIGINT
BLOB	BLOB Note 1	BLOB Note 5
	LONGVARBINARY Note 1	LONGVARBINARY Note 5
CHAR(size)	CHAR	CHAR
CHAR(size) FOR BIT DATA	BINARY	BINARY
CLOB	CLOB Note 1	CLOB Note 5
	LONGVARCHAR Note 1	LONGVARCHAR Note 5
DATE	DATE	DATE
DBCLOB	CLOB Note 1	CLOB Note 5
	LONGVARCHAR Note 1	LONGVARCHAR Note 5
DECIMAL	DECIMAL	DECIMAL
FLOAT	FLOAT	FLOAT
	DOUBLE	DOUBLE
GRAPHIC(size)	CHAR	CHAR Note 4

NOTES:

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.

Table B-7. Mapping DB2 UDB Data Types to JDBC Data Types (cont.)

DB2 SQL Data Type	JDBC Data Type Non-Unicode Service	JDBC Data Type Unicode Service
INTEGER	INTEGER	INTEGER
LONG VARCHAR	LONGVARCHAR Note 1	LONGVARCHAR Note 1
LONG VARCHAR FOR BIT DATA	LONGVARBINARY Note 1	LONGVARBINARY Note 1
LONG VARGRAPHIC	LONGVARCHAR	LONGVARCHAR Note 4
NUMERIC	NUMERIC	NUMERIC
REAL	REAL	REAL
SMALLINT	SMALLINT	SMALLINT
TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP
	CHAR Note 2	CHAR Note 2
VARCHAR(size)	VARCHAR	VARCHAR
VARCHAR(size) FOR BIT DATA	VARBINARY	VARBINARY
VARGRAPHIC(size)	VARCHAR	VARCHAR Note 4

NOTES:

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 DB2 UDB Data Types 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[]
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 Note 1	String Note 1
VARCHAR(size)	String	String
VARCHAR(size) FOR BIT DATA	byte[]	byte[]
VARGRAPHIC	String	String

NOTE

1 DataSourceFetchTimestampAsString specifies whether timestamps are fetched as character strings

Informix 9

The following table lists the data types that the ODBC driver supports for Informix 9:

Table B-9. Mapping Informix 9 Data Types to ODBC Data Types

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 9:

Table B-10. Mapping Informix 9 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 9:

Table B-11. Mapping Informix 9 Data Types to JDBC Data Types

Informix Data Type	JDBC Data Type
Blob Note 1	LONGVARBINARY Note 2
Boolean	BIT
Byte	LONGVARBINARY
Char	CHAR
Clob Note 1	LONGVARCHAR Note 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 lists the data types that the .NET data provider supports for Informix 9.

Table B-12. Mapping Informix 9 Data Types to .NET Data Types

Informix Data Type	.NET Data Type
Blob	LVARBINARY
Boolean	BIT
Byte	LVARBINARY
Char	CHAR
Clob	LVARCHAR
Date	DATE
Datetime hour to second	TIME
Datetime year to fraction(5)	TIMESTAMP
Decimal	DECIMAL
Float	DOUBLE
Integer	INT
Int8	BIGINT
Lvarchar	LVARCHAR
Money	DECIMAL
Numeric	DECIMAL
Serial	INT
Serial8	BIGINT
Smallfloat	SINGLE
Smallint	SMALLINT
Text	LVARCHAR
Varchar	VARCHAR

* The LOB data types will be handled as LOB objects or as Longvar data types, depending on the setting for the DataSourceLobAsLongvar service attribute. Refer to the *SequeLink Administrator's Guide* for more information

Microsoft SQL Server 7, 2000

The following table maps the ODBC data types that the ODBC driver supports to the Microsoft SQL Server 7 and Microsoft SQL Server 2000.

Table B-13. Mapping Microsoft SQL Server Data Types to ODBC Data Types

Microsoft SQL Server7 Data Type	Microsoft SQL Server 2000 Data Type	ODBC Data Type
--	bigint	SQL_BIGINT
--	bigint identity	SQL_BIGINT
binary	binary	SQL_BINARY
bit	bit	SQL_BIT
char	char	SQL_CHAR Note 1
		SQL_WCHAR Note 2
datetime	datetime	SQL_DATETIME Note 3
		SQL_CHAR Note 4
decimal	decimal	SQL_DECIMAL
decimal identity	decimal identity	SQL_DECIMAL
float	float	SQL_FLOAT
guid	guid	SQL_BINARY
image	image	SQL_LONGVARBINARY
int	int	SQL_INTEGER
int identity	int identity	SQL_INTEGER
money	money	SQL_DECIMAL
nchar	nchar	SQL_CHAR Note 1
		SQL_WCHAR Note 2

NOTES:

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

Table B-13. Mapping Microsoft SQL Server Data Types to ODBC Data Types *(cont.)*

Microsoft SQL Server7 Data Type	Microsoft SQL Server 2000 Data Type	ODBC Data Type
ntext	ntext	SQL_LONGVARCHAR Note 1
		SQL_WLONGVARCHAR Note 2
numeric	numeric	SQL_NUMERIC
numeric identity	numeric identity	SQL_NUMERIC
nvarchar	nvarchar	SQL_VARCHAR Note 1
		SQL_WVARCHAR Note 2
real	real	SQL_REAL
smalldatetime	smalldatetime	SQL_DATETIME Note 3
		SQL_CHAR Note 4
smallint	smallint	SQL_SMALLINT
smallint identity	smallint identity	SQL_SMALLINT
smallmoney	smallmoney	SQL_DECIMAL
--	SQLVariant	SQL_BINARY
sysname	sysname	SQL_VARCHAR Note 1
		SQL_WVARCHAR Note 2
text	text	SQL_LONGVARCHAR Note 1
		SQL_WVARCHAR Note 2
timestamp	timestamp	SQL_BINARY
tinyint	tinyint	SQL_TINYINT
tinyint identity	tinyint identity	SQL_TINYINT
--	UniqueIdentifier	SQL_BINARY
varbinary	varbinary	SQL_VARBINARY
varchar	varchar	SQL_VARCHAR Note 1
		SQL_WVARCHAR Note 2

NOTES:

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 7 and Microsoft SQL Server 2000:

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

Microsoft SQL Server7 Data Type	Microsoft SQL Server 2000 Data Type	OLE DB Data Type
--	bigint	DBTYPE_I8
--	bigint identity	DBTYPE_I8
binary	binary	DBTYPE_BYTES
bit	bit	DBTYPE_BOOL
char	char	DBTYPE_STR
datetime	datetime	DBTYPE_DBTIMESTAMP Note 2
		DBTYPE_STR Note 3
decimal	decimal	DBTYPE_NUMERIC
decimal identity	decimal identity	DBTYPE_NUMERIC
float	float	DBTYPE_R8
guid	guid	DBTYPE_GUID
image	image	DBTYPE_BYTES
int	int	DBTYPE_I4
int identity	int identity	DBTYPE_I4
money	money	DBTYPE_NUMERIC
nchar	nchar	DBTYPE_STR Note 1
ntext	ntext	DBTYPE_STR Note 1
numeric	numeric	DBTYPE_NUMERIC
numeric() identity	numeric() identity	DBTYPE_NUMERIC
nvarchar	nvarchar	DBTYPE_STR Note 1
real	real	DBTYPE_R4

1 Limited nchar,nvarchar, ntext support only

2 Only present when DataSourceFetchTimestampAsString=False

3 Only present when DataSourceFetchTimestampAsString=True

Table B-14. Mapping Microsoft SQL Server Data Types to ADO Data Types *(cont.)*

Microsoft SQL Server7 Data Type	Microsoft SQL Server 2000 Data Type	OLE DB Data Type
smalldatetime	smalldatetime	DBTYPE_DBTIMESTAMP Note 2
		DBTYPE_STR Note 3
Smallint	Smallint	DBTYPE_I2
Smallint identity	Smallint identity	DBTYPE_I2
Smallmoney	Smallmoney	DBTYPE_NUMERIC
--	SQLVariant	DBTYPE_BYTES
Sysname	Sysname	DBTYPE_STR NOTE 1
Text	Text	DBTYPE_STR
Timestamp	Timestamp	DBTYPE_BYTES
Tinyint	Tinyint	DBTYPE_UI1
Tinyint identity	Tinyint identity	DBTYPE_I1
--	Uniqueidentifier	DBTYPE_GUID
Varbinary	Varbinary	DBTYPE_BYTES
varchar	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 7 and Microsoft SQL Server 2000.

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

Microsoft SQL Server 7 Data Type	Microsoft SQL Server 2000 Data Type	JDBC Data Type
--	Bigint	BIGINT
--	Bigint Identity	BIGINT
Binary	Binary	BINARY
Bit	Bit	BIT
Char	Char	CHAR
Datetime	Datetime	TIMESTAMP Note 2
		CHAR Note 3
Decimal	Decimal	DECIMAL
Decimal identity	Decimal identity	DECIMAL
Float	Float	FLOAT
Guid	Guid	BINARY
Image	Image	LONGVARBINARY
Int	Int	INTEGER
Int identity	Int identity	INTEGER
Money	Money	DECIMAL
Nchar	Nchar	CHAR Note 1
Ntext	Ntext	LONGVARCHAR Note 1
Numeric	Numeric	NUMERIC
Numeric identity	Numeric identity	NUMERIC
Nvarchar	Nvarchar	VARCHAR Note 1
Real	Real	REAL

NOTES:

1. Full nchar, nvarchar, and ntext support requires ServiceCodePage=Database.

2 Only present when DataSourceFetchTimestampAsString=False

3 Only present when DataSourceFetchTimestampAsString=True

Table B-15. Mapping Microsoft SQL Server Data Types to JDBC Data Types *(cont.)*

Microsoft SQL Server 7 Data Type	Microsoft SQL Server 2000 Data Type	JDBC Data Type
Smalldatetime	Smalldatetime	TIMESTAMP Note 2
		CHAR Note 3
Smallint	Smallint	SMALLINT
Smallint identity	Smallint identity	SMALLINT
Smallmoney	Smallmoney	DECIMAL
--	SQLVariant	BINARY
Sysname	Sysname	VARCHAR Note 1
Text	Text	LONGVARCHAR
Timestamp	Timestamp	BINARY
Tinyint	Tinyint	TINYINT
Tinyint identity	Tinyint identity	TINYINT
--	UniqueIdentifier	BINARY
Varbinary	Varbinary	VARBINARY
Varchar	Varchar	VARCHAR

NOTES:

1. Full nchar, nvarchar, and ntext support requires ServiceCodePage=Database.

2 Only present when DataSourceFetchTimestampAsString=False

3 Only present when DataSourceFetchTimestampAsString=True

The following table lists the data types that the .NET data provider supports for Microsoft SQL Server 7 and Microsoft SQL Server 2000.

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

Microsoft SQL Server 7 Data Type	Microsoft SQL Server 2000 Data Type	.NET Data Type
--	bigint	BIGINT
--	bigint identity	BIGINT
binary	binary	BINARY
bit	bit	BIT
char	char	CHAR
datetime	datetime	TIMESTAMP Note 2
		CHAR Note 3
decimal	decimal	DECIMAL
decimal identity	decimal identity	DECIMAL
float	float	DOUBLE
guid	guid	BINARY
image	image	LVARBINARY
int	int	INT
int identity	int identity	DECIMAL
money	money	DECIMAL
nchar	nchar	CHAR Note 1
ntext	ntext	LVARCHAR Note 1
numeric	numeric	DECIMAL
numeric identity	numeric() identity	DECIMAL
nvarchar	nvarchar Note 1	VARCHAR Note 1
real	real	SINGLE

1 Full nchar,nvarchar, ntext support requires ServiceCodePage=Database

2 Only present when DataSourceFetchTimestampAsString=False

3 Only present when DataSourceFetchTimestampAsString=True

Table B-16. Mapping Microsoft SQL Server Data Types to .NET Data Types *(cont.)*

Microsoft SQL Server 7 Data Type	Microsoft SQL Server 2000 Data Type	.NET Data Type
smalldatetime	smalldatetime	TIMESTAMP Note 2
		CHAR Note 3
smallint	smallint	INT
smallint identity	smallint identity	INT
smallmoney	smallmoney	DECIMAL
--	SQLVariant	BINARY
sysname	sysname	VARCHAR Note 1
text	text	LVARCHAR
timestamp	timestamp	BINARY
tinyint	tinyint	BYTE
tinyint identity	tinyint identity	BYTE
--	uniqueidentifier	BINARY
varbinary	varbinary	VARBINARY
varchar	varchar	VARCHAR

1 Full nchar,nvarchar, ntext support requires ServiceCodePage=Database

2 Only present when DataSourceFetchTimestampAsString=False

3 Only present when DataSourceFetchTimestampAsString=True

Oracle8, Oracle9i

The following table lists the data types that the ODBC driver supports for Oracle8 and Oracle9i:

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

Oracle8 Data Type	Oracle9i Data Type	ODBC Data Type
BFILE	BFILE	SQL_LONGVARBINARY
BLOB	BLOB	SQL_LONGVARBINARY
CHAR(size)	CHAR(size)	SQL_CHAR Note 1
		SQL_WCHAR Note 2
CLOB	CLOB	SQL_LONGVARCHAR Note 5
		SQL_WLONGVARCHAR Note 2
DATE	DATE	SQL_TYPE_TIMESTAMP
LONG	LONG	SQL_LONGVARCHAR Note 1
		SQL_WLONGVARCHAR Note 2
LONG RAW	LONG RAW	SQL_LONGVARBINARY
NCHAR(size)	NCHAR(size)	SQL_CHAR Note 3
		SQL_WCHAR Note 4
NCLOB	NCLOB	SQL_LONGVARCHAR Note 3
		SQL_WLONGVARCHAR Note 4

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 national character set is Unicode

4 Only present when ServiceCodePage=Database and national character set is Unicode

5 Not present when ServiceCodePage=Database and database 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.

Table B-17. Mapping Oracle Data Types to ODBC Data Types *(cont.)*

Oracle8 Data Type	Oracle9i Data Type	ODBC Data Type
NUMBER	NUMBER	SQL_FLOAT
NUMBER(p,s)	NUMBER(p,s)	SQL_DECIMAL
NVARCHAR2(size)	NVARCHAR2(size)	SQL_VARCHAR Note 3
		SQL_WVARCHAR Note 4
RAW(size)	RAW(size)	SQL_VARBINARY
ROWID	ROWID	SQL_VARCHAR
TIMESTAMP	TIMESTAMP	SQL_TIMESTAMP Note 6
		SQL_CHAR Note 7
--	TIMESTAMP WITH LOCAL TIME ZONE	SQL_TYPE_TIMESTAMP Note 6
		SQL_CHAR Note 7
--	TIMESTAMP WITH TIME ZONE	SQL_TYPE_TIMESTAMP Note 6,8, 9
--	TIMESTAMP WITH TIME ZONE	SQL_CHAR Note 7, 8
VARCHAR2(size)	VARCHAR2(size)	SQL_VARCHAR Note 1
VARCHAR2(size)	VARCHAR2(size)	SQL_WVARCHAR Note 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 national character set is Unicode

4 Only present when ServiceCodePage=Database and national character set is Unicode

5 Not present when ServiceCodePage=Database and database 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 Oracle8 and Oracle9i:

Table B-18. Mapping Oracle Data Types to ADO Data Types

Oracle8 Data Type	Oracle9i Data Type	OLE DB Data Type
BFILE	BFILE	DBTYPE_BYTES
BLOB	BLOB	DBTYPE_BYTES
CHAR(size)	CHAR(size)	DBTYPE_STR
CLOB	CLOB	DBTYPE_STR
DATE	DATE	DBTYPE_DBTIMESTAMP
LONG	LONG	DBTYPE_STR
LONG RAW	LONG RAW	DBTYPE_BYTES
NCHAR(size)	NCHAR(size)	DBTYPE_STR Note 1
NCLOB	NCLOB	DBTYPE_STR Note 1
NUMBER	NUMBER	DBTYPE_R8
NUMBER(P,S)	NUMBER(p,s)	DBTYPE_NUMERIC
NVARCHAR2(size)	NVARCHAR2(size)	DBTYPE_STR Note 1
RAW(size)	RAW(size)	DBTYPE_BYTES
ROWID	ROWID	DBTYPE_STR
--	TIMESTAMP	DBTYPE_DBTIMESTAMP Note 2
		DBTYPE_STR Note 3
--	TIMESTAMP WITH LOCAL TIME ZONE	DBTYPE_DBTIMESTAMP Note 2
		DBTYPE_STR Note 3

NOTES

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.

Table B-18. Mapping Oracle Data Types to ADO Data Types (cont.)

Oracle8 Data Type	Oracle9i Data Type	OLE DB Data Type
--	TIMESTAMP WITH TIME ZONE	DBTYPE_DBTIMESTAMP Note 2, 4, 5
		DBTYPE_STR Note 3, 4
--	VARCHAR2	DBTYPE_STR

NOTES

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 Oracle8 and Oracle9i:

Table B-19. Mapping Oracle Data Types to JDBC Data Types

Oracle8 Data Type	Oracle9i Data Type	JDBC Data Type
BFILE	BFILE	LONGVARBINARY
BLOB	BLOB	BLOB Note 1
		LONGVARBINARY Note 2
CHAR(size)	CHAR(size)	CHAR

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.

Table B-19. Mapping Oracle Data Types to JDBC Data Types *(cont.)*

Oracle8 Data Type	Oracle9i Data Type	JDBC Data Type
CLOB	CLOB	CLOB Note 1
		LONGVARCHAR Note 2
DATE	DATE	TIMESTAMP
LONG	LONG	LONGVARCHAR Note 1
LONG RAW	LONG RAW	LONGVARBINARY Note 1
NCHAR(size)	NCHAR(size)	CHAR
NCLOB	NCLOB	CLOB Note 1
		LONGVARCHAR Note 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 Note 3
		CHAR Note 4
--	TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP Note 3
		CHAR Note 4
--	TIMESTAMP WITH TIME ZONE	TIMESTAMP Note 3, 5, 6
		CHAR Note 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 Oracle8 and Oracle9i:

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

Oracle8 Data Type	Oracle9i Data Type	.NET Data Type
BFILE	BFILE	LVARBINARY
BLOB	BLOB	BLOB
CHAR(size)	CHAR(size)	CHAR
CLOB	CLOB	CLOB
DATE	DATE	TIMESTAMP
LONG	LONG	LVARCHAR
LONG RAW	LONG RAW	LVARBINARY
NCHAR(size)	NCHAR(size)	CHAR
NCLOB	NCLOB	CLOB
NUMBER	NUMBER	DOUBLE
NUMBER(P,S)	NUMBER(p,s)	DECIMAL
NVARCHAR2(size)	NVARCHAR2(size)	VARCHAR
RAW(size)	RAW(size)	LVARBINARY
ROWID	ROWID	VARCHAR
--	TIMESTAMP	TIMESTAMP Note 1
		CHAR Note 2
--	TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP Note 1
		CHAR Note 2
--	TIMESTAMP WITH TIME ZONE	TIMESTAMP Note 1, 3, 4
		CHAR Note 2, 3
--	VARCHAR2(size)	VARCHAR

1 Only present when DataSourceFetchTimestampAsString=False

2 Only present when DataSourceFetchTimestampAsString=True

3 Only present when DataSourceORAMapTSWTZ=True

4 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.

Sybase

The following table lists the data types that the ODBC driver supports for Sybase 11 and 12:

Table B-21. Mapping Sybase Data Types to ODBC Data Types

Sybase Data Type	ODBC Data Type
binary	SQL_BINARY
bit	SQL_BIT
char	SQL_CHAR Note 1
	SQL_WCHAR Note 2
datetime	SQL_TIMESTAMP Note 3
	SQL_CHAR Note 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 Note 3
	SQL_CHAR Note 4
smallint	SQL_SMALLINT

NOTES

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

Table B-21. Mapping Sybase Data Types to ODBC Data Types *(cont.)*

Sybase Data Type	ODBC Data Type
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR Note 5
	SQL_WVARCHAR Note 2
text	SQL_LONGVARCHAR Note 1
	SQL_WLONGVARCHAR Note 2
timestamp	SQL_BINARY
tinyint	SQL_TINYINT
unichar	SQL_WCHAR Note 6
univarchar	SQL_WVARCHAR Note 6
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR Note 1
	SQL_WVARCHAR Note 2

NOTES

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 11 and 12:

Table B-22. Mapping Sybase Data Types to ADO Data Types

Sybase Data Type	OLE DB Data Type
binary	DBTYPE_BYTES
bit	DBTYPE_BOOL
char	DBTYPE_STR
datetime	DBTYPE_DBTIMESTAMP Note 1
	DBTYPE_STR Note 2
decimal	DBTYPE_NUMERIC
float	DBTYPE_R8
image	DBTYPE_BYTES
int	DBTYPE_I4
money	DBTYPE_CY
numeric	DBTYPE_R4
real	DBTYPE_R4
smalldatetime	DBTYPE_DBTIMESTAMP Note 1
	DBTYPE_STR Note 2
smallint	DBTYPE_I2
smallmoney	DBTYPE_CY
sysname	DBTYPE_STR
text	DBTYPE_STR
timestamp	DBTYPE_BYTES
tinyint	DBTYPE_UI1
varbinary	DBTYPE_BYTES
varchar	DBTYPE_STR

NOTES

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 11 and 12:

Table B-23. Mapping Sybase Data Types to JDBC Data Types

Sybase Data Type	JDBC Data Type
binary	BINARY
bit	BIT
char	CHAR
datetime	TIMESTAMP Note 1
	CHAR Note 2
decimal	DECIMAL
float	FLOAT
image	LONGVARBINARY
int	INTEGER
money	DECIMAL
numeric	NUMERIC
real	REAL
smalldatetime	TIMESTAMP Note 1
	CHAR Note 2
smallint	SMALLINT
smallmoney	DECIMAL
sysname	VARCHAR
text	LONGVARCHAR
timestamp	BINARY
tinyint	TINYINT
unichar	CHAR Note 3

NOTES

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

Table B-23. Mapping Sybase Data Types to JDBC Data Types *(cont.)*

Sybase Data Type	JDBC Data Type
univarchar	VARCHAR Note 3
varbinary	VARBINARY
varchar	VARCHAR

NOTES

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 11 and 12:

Table B-24. Mapping Sybase Data Types to .NET Data Types

Sybase Data Type	.NET Data Type
binary	BINARY
bit	BIT
char	CHAR
datetime	TIMESTAMP Note 1
	CHAR Note 2
decimal	DECIMAL
float	DOUBLE
image	LVARBINARY
int	INT

NOTES

1 Only present when DataSourceFetchTimestampAsString=False

2 Only present when DataSourceFetchTimestampAsString=True

Table B-24. Mapping Sybase Data Types to .NET Data Types *(cont.)*

Sybase Data Type	.NET Data Type
money	DECIMAL
numeric	DECIMAL
real	SINGLE
smalldatetime	TIMESTAMP Note 1
	CHAR Note 2
smallint	SMALLINT
smallmoney	DECIMAL
sysname	VARCHAR
text	LVARCHAR
timestamp	BINARY
tinyint	BYTE
unichar	CHAR
univarchar	VARCHAR
varbinary	VARBINARY
varchar	VARCHAR

NOTES

1 Only present when DataSourceFetchTimestampAsString=False

2 Only present when DataSourceFetchTimestampAsString=True

Isolation Levels

This section lists the isolation levels supported by SequeLink for each data store, including their default isolation level.

Table B-25. Isolation Levels

Database	Isolation Levels	Default
DB2 on OS/390 V5, V6, V7	Read uncommitted Read committed Repeatable read Serializable	Read committed
DB2 UDB V7.2, V8.1	Read uncommitted Read committed Repeatable read Serializable	Read committed
Informix	Read uncommitted Read committed Repeatable read	Read committed
Microsoft SQL Server	Read uncommitted Read committed Repeatable read Serializable	Read committed
Oracle8	Read committed Serializable	Read committed
Oracle9i	Read committed Serializable	Read committed
Sybase	Read uncommitted Read committed Repeatable read Serializable	Read committed

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 1.2 to 1.4, and the corresponding JDBC level.

Supported Functionality

The following tables list functionality supported for each JDBC object.

Array Object **Methods**	**Version Introduced**	**Supported**	**Comments**
(all)	2.0 Core	No	Array objects are neither exposed, nor taken as input.

Blob Object **Methods**	**Version Introduced**	**Supported**	**Comments**
InputStream getBinaryStream ()	2.0 Core	Yes	
NOTE: Blob support is emulated using LONGVARBINARY datatypes on DB2 z/OS V5, SQL Server, Sybase, and ODBC Socket. Using emulated Blobs incurs a performance and scalability penalty.			

Blob Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 DB2 z/OS V5, SQL Server, Sybase, and ODBC Socket. Using emulated Blobs incurs a performance and scalability penalty.			

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.
BigDecimal getBigDecimal (int)	2.0 Core	Yes	
BigDecimal getBigDecimal (int, int)	1.0	Yes	
BigDecimal getBigDecimal (String)	3.0	No	Throws “unsupported method” exception.

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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.

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
int executeUpdate (String, int)	3.0	No	Throws “invalid method call” exception.
int executeUpdate (String, int[])	3.0	No	Throws “invalid method call” exception.

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
Object getObject (String, Map)	3.0	No	Throws “unsupported method” exception.
ParameterMetaData getParameterMetaData()	3.0	Yes	Not supported on Oracle or Informix Servers.

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
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	

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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.

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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	

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	SQLServer throws "optional feature not implemented" exception.
void setDate (int, Date, Calendar)	2.0 Core	Yes	SQLServer 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.
void setDouble (int, double)	1.0	Yes	
void setDouble (String, double)	3.0	No	Throws “unsupported method” exception.
void setEscapeProcessing (boolean)	1.0	Yes	

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	SQLServer 7 throws "optional feature not implemented" exception. However, you can use 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	
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).

CallableStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 ODBC Socket Server is dependent on the ODBC driver it loads.
void setRef (int, Ref)	2.0 Core	No	Throws “unsupported method” exception.
void setShort (int, short)	1.0	Yes	
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.

CallableStatement Object *(cont.)* Methods	Version Introduced	Supported	Comments
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 Methods	Version Introduced	Supported	Comments
InputStream getAsciiStream ()	2.0 Core	Yes	
NOTE: Clob support is emulated using LONGVARCHAR datatypes on DB2 z/OS V5, SQL Server, Sybase, and ODBC Socket. Using emulated Clobs incurs a performance and scalability penalty.			

Clob Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 DB2 z/OS V5, SQL Server, Sybase, and ODBC Socket. Using emulated Clobs incurs a performance and scalability penalty.			

Connection Object **Methods**	**Version Introduced**	**Supported**	**Comments**
void clearWarnings ()	1.0	Yes	
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	

Connection Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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

Connection Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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	

Connection Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
void setSavepoint ()	3.0	Yes	Not supported with DB2 V5 on z/OS, DB2 V5.2 and DB2 V6.1, and Informix servers.
void setSavepoint (String)	3.0	Yes	Not supported with DB2 V5 on z/OS, DB2 V5.2 and DB2 V6.1, and Informix servers.
void setTransactionIsolation (int)	1.0	Yes	
void setTypeMap (Map)	2.0 Core	Yes	Ignored.

ConnectionPoolData Source 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	
void setLogWriter (PrintWriter)	2.0 Optional	Yes	
void setLoginTimeout (int)	2.0 Optional	Yes	

DatabaseMetaData Object Methods	Version 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.
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	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
int getResultSetHoldability ()	3.0	Yes	
ResultSet getSchemas ()	1.0	Yes	The JDBC driver supports this method if the data store has support for it.

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 has support for it.
ResultSet getTableTypes ()	1.0	Yes	
ResultSet getTables (String, String, String, String [])	1.0	Yes	
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	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
ResultSet getUDTs (String, String, String, int [])	2.0 Core	No	SequeLink does not support the Advanced Data Type functionality and always returns and 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 and 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 has support for 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	
boolean nullsAreSortedAtEnd ()	1.0	Yes	
boolean nullsAreSortedAtStart ()	1.0	Yes	
boolean nullsAreSortedHigh ()	1.0	Yes	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
boolean supportsBatchUpdates ()	2.0 Core	Yes	
boolean supportsCatalogsInData Manipulation ()	1.0	Yes	
boolean supportsCatalogsInIndex ()	1.0	Yes	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
boolean supportsGroupByBeyondSelect ()	1.0	Yes	
boolean supportsGroupByUnrelated ()	1.0	Yes	
boolean supportsIntegrityEnhancement Facility ()	1.0	Yes	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
boolean supportsOpenStatementsAcross Rollback ()	1.0	Yes	
boolean supportsOrderByUnrelated ()	1.0	Yes	
boolean supportsOuterJoins ()	1.0	Yes	

DatabaseMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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	

DatabaseMetaData Object *(cont.)* Methods	Version Introduced	Supported	Comments
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 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.

DataSource Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
void setLoginTimeout (int)	2.0 Optional	Yes	

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 **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	

ParameterMetaData Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
int isNullable (int)	3.0	Yes	
boolean isSigned (int)	3.0	Yes	

PooledConnection Object **Methods**	**Version Introduced**	**Supported**	**Comments**
void addConnectionEventListener (ConnectionEventListener)	2.0 Optional	Yes	
void close()	2.0 Optional	Yes	
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 occurrance). The drivers do not support this "reclaiming" of connections and will throw a SQLException "Reclaim of open connection is not supported."

PooledConnection Object *(cont.)* Methods	Version Introduced	Supported	Comments
void removeConnectionEvent Listener (ConnectionEventListener)	2.0 Optional	Yes	

PreparedStatement Object Methods	Version Introduced	Supported	Comments
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	
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.

PreparedStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
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	

PreparedStatement Object *(cont.)* Methods	Version Introduced	Supported	Comments
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	
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	

PreparedStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
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	SQL Server 7 throws "optional feature not implemented" exception. However, you can us the workaround MSSMapLongToDecimal to map setLong to DECIMAL instead of BIGINT.
void setMaxFieldSize (int)	1.0	Yes	
void setMaxRows (int)	1.0	Yes	
void setNull (int, int)	1.0	Yes	

PreparedStatement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	Not supported on DB2 z/OS servers. Support for ODBC Socket Server is dependent on the ODBC 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	
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.

PreparedStatement Object *(cont.)* Methods	Version Introduced	Supported	Comments
void setURL (int, URL)	3.0	No	Throws “unsupported method” exception.
boolean wasNull ()	1.0	Yes	

Ref Object Methods	Version Introduced	Supported	Comments
(all)	2.0 Core	No	Ref objects are neither exposed, nor taken as input.

Referenceable Object Methods	JDBC Version Introduced	Supported	Comments
Reference getReference()	javax.naming	Yes	Implemented by SequeLinkDataSource.

ResultSet Object Methods	Version Introduced	Supported	Comments
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	

ResultSet Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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.
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 hexidecimal 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 *(cont.)* **Methods**	**Version 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 hexidecimal representation and returns the ASCII representation. For example, the binary value 20 is represented in hexidecimal 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 *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 *(cont.)* **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 *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
boolean isBeforeFirst ()	2.0 Core	Yes	
boolean isFirst ()	2.0 Core	Yes	
boolean isLast ()	2.0 Core	Yes	
boolean last ()	2.0 Core	Yes	
void moveToCurrentRow ()	2.0 Core	Yes	
void moveToInsertRow ()	2.0 Core	Yes	
boolean next ()	1.0	Yes	
boolean previous ()	2.0 Core	Yes	
void refreshRow ()	2.0 Core	Yes	
boolean relative (int)	2.0 Core	Yes	
boolean rowDeleted ()	2.0 Core	Yes	
boolean rowInserted ()	2.0 Core	Yes	
boolean rowUpdated ()	2.0 Core	Yes	
void setFetchDirection (int)	2.0 Core	Yes	
void setFetchSize (int)	2.0 Core	Yes	
void updateArray (int, Array)	3.0	No	Throws “unsupported method” exception.
void updateArray (String, Array)	3.0	No	Throws “unsupported method” exception.
void updateAsciiStream (int, InputStream, int)	2.0 Core	Yes	
void updateAsciiStream (String, InputStream, int)	2.0 Core	Yes	
void updateBigDecimal (int, BigDecimal)	2.0 Core	Yes	
void updateBigDecimal (String, BigDecimal)	2.0 Core	Yes	

ResultSet Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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 **Methods**	**Version Introduced**	**Supported**	**Comments**
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 getColumnTypeName (int)	1.0	Yes	
int getPrecision (int)	1.0	Yes	
int getScale (int)	1.0	Yes	

ResultSetMetaData Object *(cont.)* Methods	Version Introduced	Supported	Comments
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 Methods	Version Introduced	Supported	Comments
(all)	2.0 Optional	No	

SavePoint Object Methods	Version Introduced	Supported	Comments
int getSavepointId ()	3.0	Yes	Savepoint not supported on Informix and ODBC Socket Server servers.

SavePoint Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
String getSavepointName ()	3.0	Yes	Savepoint not supported on Informix and ODBC Socket Server servers.

Serializable Object **Methods**	**Version Introduced**	**Supported**	**Comments**
(N/A)	java.io	Yes	Implemented by SequeLinkDataSource.

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	
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.

Statement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	
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 and Informix.
ResultSet getResultSet ()	1.0	Yes	

Statement Object *(cont.)* **Methods**	**Version Introduced**	**Supported**	**Comments**
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	Not supported on DB2 z/OS servers. Support for ODBC Socket Server is dependent on the ODBC driver it loads.

Struct Object **Methods**	**Version Introduced**	**Supported**	**Comments**
(all)	2.0	No	Struct objects are neither exposed, nor taken as input.

XAConnection Object Methods	Version Introduced	Supported	Comments
XAResource getXAResource()	2.0 Optional	Yes	

XADataSource Object Methods	Version Introduced	Supported	Comments
int getLoginTimeout ()	2.0 Optional	Yes	
PrintWriter getLogWriter ()	2.0 Optional	Yes	
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

Establishing JDBC connections is resource-expensive, especially when the JDBC API is used in a middle-tier server environment. In this type of environment, 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.

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 271 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 471), 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 468. 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 466).

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

Whether connection pooling is used does not affect application code. It does not require any code changes to the application because the application performs a lookup on a JNDI name of a previously registered data source. If the data source specifies a connection pooling implementation during JNDI registration (as described in "Creating a Data Source Using the DataDirect Connection Pool Manager" on page 468), the client application benefits from faster connections through connection pooling.

The following example shows code that can be used to look up and use a 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 468).

```
//*******************************************************************
//
// 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 466 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
```

Terminating the Pool Manager

The Pool Manager requires notification of application termination to close connections in the pool properly. 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 .NET Code Examples

This appendix includes code examples of typical database access tasks in ADO.NET. 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".
- To create the tables on Sybase, use the script in "Sample Tables for Sybase" on page 477.

Sample Tables for Oracle

The following script can be run in SQL*Plus. See 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. See 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 477).

```
// 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 475).

```
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-2002',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 475), 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.

```
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 commited 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-2002',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 475).

```
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 477).

```
// 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 475).

```
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 309 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 475).

```
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 477).

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 477), 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 338
  Framework data types for .NET data provider 320
  security 345
  supported public objects 326
.NET Client
  assembly name 308
  connecting to a database 314
  connection string attributes 306
  creating a connection pool 303
  data types 320
  data types supported
    DB2 on OS/390 378
    DB2 UDB 387
    Informix 389
    Microsoft SQL Server 393
    Oracle8 401
    Oracle9i 401, 406
    Sybase 407
  database errors 347
  event handling 325
  isolation levels 325
  mapping parameter types 321
  object
    InfoMessageEventArgs 341
    SequeLinkCommand 334
    SequeLinkCommandBuilder 336
    SequeLinkConnection 337
    SequeLinkDataAdapter 338
    SequeLinkDataReader 338
    SequeLinkError 339
    SequeLinkErrorCollection 340
    SequeLinkException 340
    SequeLinkTrace 343
    SequeLinkTransaction 344
  online help 21
  parameter arrays 309
  parameter markers 309
  provider-specific methods supported 333
  provider-specific properties supported 333
  SequeLink client errors 346
  setting a savepoint 344
  thread support 325
  tracing method calls 313
  transaction support 311
? for parameter markers 309

Numerics

64-bit ODBC Client (UNIX)
  configuring 46
  example 47

A

ABS function 369
Access Order 154, 167
accessor support 141, 162
ACOS function 369
Active Sessions 158
adding connections to a connection pool 304
ADO
  Command object 154
  Connection object 157
  mapping OLE DB methods 154
  OLE DB interfaces supported 152
  Recordset object 165

ADO Client
copying data sources 116
creating data sources 112
deleting data sources 116
modifying data sources 115
renaming data sources 115
ADO client data sources. *See* client data sources
ADO.NET
namespace 319
use of OLE DB 348
AllowPartiallyTrustedCallers 345
API functions 65
application developers
.NET 319
ADO 135
JDBC 271
ODBC 64
application ID
generating automatically 79, 173
specifying explicitly 78, 130, 173
specifying for ADO provider 173
specifying for JDBC driver 279
ApplicationID ODBC connection attribute 50
ApplicationName
ADO Client 130
JDBC Client 201
ODBC Client 50
arguments, null 85
Array object 415
arrays, parameter 309
ASCII
converting to character 366
function 366
ASIN function 369
assembly name of the .NET provider 308
Asynchable
Abort 158
Commit 158
ATAN function 369
ATAN2 function 369
attributes
ADO 130
DistinguishedName 133
ODBC 50
Autocommit Isolation Levels 158
AutomaticApplicationID ODBC connection attribute 51

B

BigDecimal objects 284
blanks, generating 368
Blob object 415
BlockFetchForUpdate 51
Blocking Storage Objects 154, 167
Bookmark Information 167
Bookmark Type 167
bookmarks 148
bound columns 92

C

CallableStatement object 416
catalog functions
defined 84
effect on performance 85
Catalog Location 158
Catalog Term 158
Catalog Usage 159
CEILING function 369
centralized system information file 48
Change Inserted Rows 154, 167
changing directories for ADO client data sources 117
CHAR function 366
cipher suites 203

Client
  SequeLink for .NET 301
  SequeLink for ADO 105
  SequeLink for JDBC 179
  SequeLink for ODBC 29
client data sources
  configuring
    ODBC file 34
    ODBC User and System 31
    on UNIX 46
  creating
    ADO 112
    ODBC 30
  deleting ADO 116
  modifying ADO 115
client errors, SequeLink (.NET) 346
Clob object 426
closing opened connections (.NET) 338
code examples
  calling a stored procedure 489
  limiting the rows returned by a Select statement 486
  retrieving a scalar value 491
  retrieving warning information 490
  updating data in a DataSet 487
  using a distributed transaction 481
  using a local transaction with a DataReader 479
  using the CommandBuilder 483
code page support for ODBC Client 33, 60
Column Definition 159
Column Privileges 155, 167
COM Object Support 159
Command object
  dynamic properties 154
  methods 154
  threading support 325
Command object (.NET)
  calling stored procedures 489
  properties 334
CommandBuilder object
  code example 483
  impact on performance 349
  properties 336
committing data 98
compatibility of JDBC versions 415
compiler requirements, UNIX 65
CONCAT function 367
concurrency types for result sets 277
Configuration Manager
  menu bar 107
  overview 106
configuring
  ADO client data sources 111
  file client data sources 34
  JDBC data sources 190
  ODBC client data source for UNIX 46
connecting
  improving performance 352
  overview 302
  resetting the state of the connection 306
  specifying connection properties 305
  to data source, logon dialog box 121
  using DataDirect Test 215
  using URLs (JDBC) 188
  with a provider string 128
  with JDBC data sources 200
connection
  attributes in ADO 130
  attributes in ODBC 50
  defining using master data source file 120
  handles 98
  JDBC options 179
  managing 97
  non-pooled 473
  ODBC options 29
  precedence for JDBC driver 200
  properties for JDBC driver 200
  testing ADO 121
  testing ODBC on Windows 45
connection attributes, specifying 305
Connection dialog box 121
Connection Lifetime 306
Connection object (.NET)
  properties 337
  threading 325
  using in the namespace 319

Connection object (ADO)
dynamic properties 158
methods supported 157
Connection object (JDBC) 427
connection pool manager, about 182
connection pooling
adding connections (.NET) 304
connecting to a data source 471
Connection Pool Manager 465
creating 303
creating a data source 468
creating a pooled data source object 466
enabling 307
JDBC driver 192
maximum number of pools 307
minimum number of connections 307
ODBC driver 98
removing connections (.NET) 304
terminating the pool manager 474
using 302
Connection Status 159
contacting Technical Support 24
conventions used in this book 22
converting variant types to date/time types 172
copying, ADO client data sources 116
COS function 369
COT function 369
creating
ADO client data sources 112
connection string for .NET provider 305
JDBC data sources 191
master data source file 120
ODBC Client data sources 30
template data source file 119
CURDATE function 371
Current Catalog 159
cursors
forward-only 338
forward-only result sets (JDBC) 276
keyset-driven scrollable (ODBC) 72
scroll-insensitive (Java 2 platform) 209
scroll-insensitive result sets (JDBC) 276
scroll-sensitive result sets (JDBC) 276
static scrollable (ODBC) 72
support for scrollable (JDBC) 277
support for scrollable (ODBC) 72
using scrollable instead of cursor library (ODBC) 96
CURTIME function 371

D

data shaping 170
data source
configuring
JDBC 190
on UNIX (ODBC) 46
User and System (ODBC) 31
values overridden 171
connecting with JDBC 200
creating
ADO client 112
JDBC 191
non-pooled 466
ODBC file client 34
ODBC system 31
ODBC user 31
pooled 466
default 118
displaying properties 109
examples of JDBC 191
file
creating a master 120
creating a template 119
non-pooled 466
system 31
user 31
data source information property group 140
Data Source Name 159
Data Source Object Threading Model 159
Data Source property 159
data source property group 140

data types
.NET 320
choosing to improve performance 350
DB2 on OS/390 378
DB2 UDB 382
Informix 9 389
mapping year format 172
Microsoft SQL Server 393
Oracle8, Oracle9i 401, 406
Sybase 407
DataAdapter object
code example 484
properties 338
database
data dictionary 88
meta-information, retrieving 88
naming using JNDI 191
Database connection attribute
.NET Client 306
ADO provider 131
ODBC driver 51
Database Data Dictionary
filters 89
views (DB2 for z/OS) 89
database errors (.NET client) 347
DATABASE function 373
DatabaseMetaData object 431
database-specific information 377
DataDirect Configuration Manager
menu bar 107
DataDirect Connection Pool Manager 185
DataDirect Spy
attributes 265
jar file 186
log example 267
overview 181
registering the JDBC driver 264
URL syntax 265
using with JDBC data sources 266
DataDirect Test
batch execution on a prepared statement 233
configuring 212
connecting 215
establishing savepoints 238
executing a prepared statement 223
executing a simple select statement 221
files 185
LOB support 257
retrieving database metadata 227
returning ParameterMetaData 237
scrolling through a result set 229
starting 213
tutorial 211
updateable result sets 245
using 211
DataReader object
code example 478, 479
importance of closing 338
properties 338
DataSet
code example 487
creating and using 354
DataSource object 441
date
returning current 371
returning month 372
returning year 372
scalar functions 370
DAYNAME function 371
DAYOFMONTH function 371
DAYOFWEEK function 371
DAYOFYEAR function 371
DB2 for OS/390
data types supported 378
Database Data Dictionary views 89
Distributed Transaction Management support 195
support for scrollable cursors (JDBC) 277
support for scrollable cursors (ODBC) 72
DB2 UDB
data types 382
support for scrollable cursors (ODBC) 72
DBMS Name property 159
DBMS Version property 159
DBPassword 51
DBPROP_
ROWSETCONVERSIONSONCOMMAND 144

DBPROP_ SUBQUERIES 145
DBPROP_ SUPPORTEDTXNDDL 145
DBPROP_ SUPPORTEDTXNISOLEVELS 145
DBPROP_ SUPPORTEDTXNISORETAIN 145
DBPROP_ABORTPRESERVE 148
DBPROP_ACCESSORDER 148
DBPROP_ASYNCTXNABORT 141
DBPROP_ASYNCTXNCOMMIT 141
DBPROP_AUTH_ PASSWORD 146
DBPROP_AUTH_ USERID 146
DBPROP_AUTH_PASSWORD 146
DBPROP_AUTH_PERSIST_SENSITIVE_AUTHIN FO 146
DBPROP_AUTH_USERID 146
DBPROP_BLOCKINGSTORAGEOBJECTS 148
DBPROP_BOOKMARKINFO 148
DBPROP_BOOKMARKS 148
DBPROP_BOOKMARKSKIPPED 148
DBPROP_BOOKMARKTYPE 148
DBPROP_CANFETCHBACKWARDS 148
DBPROP_CANSCROLLBACKWARDS 148
DBPROP_CATALOGLOCATION 141
DBPROP_CATALOGTERM 141
DBPROP_CATALOGUSAGE 141
DBPROP_COLUMNDEFINITION 141
DBPROP_COMMITPRESERVE 149
DBPROP_CONCATNULLBEHAVIOR 142
DBPROP_CONNECTIONSTATUS 142
DBPROP_CURRENTCATALOG 140
DBPROP_DATASOURCENAME 142
DBPROP_DATASOURCEREADONLY 142
DBPROP_DBMSNAME 142
DBPROP_DBMSVER 142
DBPROP_DELAYSTORAGEOBJECTS 149
DBPROP_DSOTHREADMODEL 142
DBPROP_HETEROGENEOUSTABLES 142
DBPROP_IDENTIFIERCASE 142
DBPROP_IMMOBILEROWS 149
DBPROP_INIT_ DATASOURCE 146
DBPROP_INIT_ HWND 146
DBPROP_INIT_ MODE 146
DBPROP_INIT_ PROMPT 146
DBPROP_INIT_ PROVIDERSTRING 147
DBPROP_INIT_CATALOG 146
DBPROP_INIT_DATASOURCE 146
DBPROP_INIT_HWND 146
DBPROP_INIT_LCID 146
DBPROP_INIT_MODE 146
DBPROP_INIT_OLEDBSERVICES 146
DBPROP_INIT_PROMPT 146
DBPROP_INIT_PROVIDERSTRING 146
DBPROP_LITERALBOOKMARKS 149
DBPROP_LITERALIDENTITY 149
DBPROP_LOCKMODE 150
DBPROP_MAXINDEXSIZE 142
DBPROP_MAXOPENROWS 150
DBPROP_MAXPENDINGROWS 150
DBPROP_MAXROWS 150
DBPROP_MAXROWSIZE 142
DBPROP_MAXROWSIZEINCLUDESBLOB 143
DBPROP_MAXTABLEINSELECT 143
DBPROP_MEMORYUSAGE 150
DBPROP_MULTIPLEPARAMSETS 143
DBPROP_MULTIPLERESULTS 143
DBPROP_MULTIPLESTORAGEOBJECTS 143
DBPROP_MULTITABLEUPDATE 143
DBPROP_NULLCOLLATION 143
DBPROP_OLEOBJECTS 143
DBPROP_OPENROWSETSUPPORT 143
DBPROP_ORDERBYCOLUMNSINSELECT 143
DBPROP_OTHERINSERT 150
DBPROP_OTHERUPDATEDELETE 150
DBPROP_OUTPUTPARAMETERAVAILABILITY 143
DBPROP_OWNINSERT 150
DBPROP_OWNUPDATEDELETE 150
DBPROP_PERSISTENTIDTYPE 143
DBPROP_PREPAREABORTBEHAVIOR 144
DBPROP_PREPARECOMMITBEHAVIOR 144
DBPROP_PROCEDURETERM 144
DBPROP_PROVIDERFRIENDLYNAME 144
DBPROP_PROVIDERNAME 144
DBPROP_PROVIDEROLEDBVER 144
DBPROP_PROVIDERVER 144
DBPROP_QUOTEDIDENTIFIERCASE 144
DBPROP_REENTRANTEVENTS 150
DBPROP_REMOVEDELETED 150
DBPROP_REPORTMULTIPLECHANGES 150

DBPROP_RETURNPENDINGINSERTS 150
DBPROP_ROWRESTRICT 151
DBPROP_ROWTHREADMODEL 151
DBPROP_SCHEMATERM 144
DBPROP_SCHEMAUSAGE 144
DBPROP_SERVERCURSOR 151
DBPROP_SERVERNAME 144
DBPROP_SESS_ AUTOCOMMITISOLEVELS 152
DBPROP_SQLSUPPORT 144
DBPROP_STRONGIDENTITY 151
DBPROP_STRUCTUREDSTORAGE 144
DBPROP_TABLETERM 145
DBPROP_TRANSACTEDOBJECT 151
DBPROP_UNIQUEROWS 151
DBPROP_UPDATABILITY 151
DBPROP_USERNAME 145
dead connections 338
default data source files
  master 118
  template 118
Default Length for Long Data 131
defining
  data source using the ODBC Administrator 31
  default setup options 118
DEGREES function 369
Delay Storage Object Updates 167
deletes, positional 99
deleting ADO client data sources 116
developing ODBC applications 64
DIFFERENCE function 367
directory structure for the JDBC Client 184
disconnected DataSet 353
Distinguished Name identifier 131
DistinguishedName attribute
  ADO 133
  ODBC 51
Distributed Transaction Management 195
distributed transactions
  code example 481
  for more information 354
documentation, SequeLink 20
downloading applets, tips 283
DPROP_GROUPBY 142
Driver object 442
DSN connection attribute 51
dynamic properties
  Command object (ADO) 154
  Connection object (ADO) 158
  Recordset object (ADO) 167

E

EnableDescribeParam 52
environment variables (UNIX) 47
error handling
  .NET data provider 345
  ADO data provider 174
  JDBC driver 281
  ODBC driver 82
Error object (.NET) 339
ErrorCollection object
  .NET data provider 340
  checking for multiple errors 340
event handling for the .NET data provider 325
examples
  64-bit ODBC Client configured for Solaris 47
  connection pooling 192
  creating and using JDBC data sources 191
  odbc.ini file configured for Solaris 46, 47
  Spy log 267
  URL (Spy) 267
Exception object (.NET) 340
EXP function 369
EXTRACT function 371

F

fetching
  backward 148, 155, 167
  BigDecimal objects 284
FetchNextOnly 52
file
  creating a master data source 120
  creating a template data source 119
file client data sources, configuring 34
File System JNDI Provider 187
files in the JDBC Client directory 184
filters, Database Data Dictionary 89
FixCharTrim 53
FLOOR function 369
forward only cursors 93
forward-only result sets (JDBC) 276
Framework types (.NET) 320
FullTrust 345
functions
  cancelling in multithreaded
    applications 71, 274
  date and time 370
  numeric 368
  ODBC API 65
  scalar 359
  string 366

G

generating ODBC application IDs
  automatically 79
GetOutputParams 53
GetSchemaTable method 330
Global Assembly Cache (GAC) 308
GROUP BY Support 160

H

header files and libraries, platform-
  specific 64
help
  .NET Client 21
  JDBC Client 21
  ODBC Client 21
help, online 21
Heterogeneous Table Support 160
HLogonID 53
Hold Rows 155, 167
Host attribute 54, 131
host password 53
Host Password attribute 131
HOUR function 371
HPassword 53

I

Identifier Case Sensitivity 160
IFNULL function 373
Immobile Rows 168
Informix 9
  data types 389
  persisting result set as XML 80
  support for scrollable cursors (ODBC) 72
Initial Catalog 160
initialization properties 145
INSERT function 367
interfaces supported by ADO provider 136
IOpenRowset 171
IPersistFile 171
IRowset interface 136
IRowsetIdentity 147
isolation levels
  .NET data provider 325
  ADO data provider 161
  by data store 413
  JDBC driver 273
  ODBC driver 69

Isolation Retention 161

J

JAR files 283
Java 2 Platform 208
Java Transaction API 195
Java Virtual Machine versions 415
JCA
  resource adapter class 273
  resource adapters
    overview 196
    using from an application 198
    using with application server 197
JDBC
  compatibility 415
  connection options 179
  forward-only result sets 276
  functionality supported 415
  scroll-insensitive result sets 276
  scroll-sensitive result sets 276
  support 415
  support for 2.0 functionality 272
JDBC 2.0 Optional Package 184, 186
JDBC Client
  configuring JDBC data sources 191
  directory structure 184
  overview 179
  using connection pooling 192
  using on a Java 2 Platform 208
  using the JTA 195
JDBC data sources 184
JDBC driver
  configuring data sources 190
  connection properties 200, 201
  creating data sources 191
  data types supported
    DB2 on OS/390 378
    DB2 UDB 386
    Informix 389
    Microsoft SQL Server 393
    Oracle8 401
    Oracle9i 401, 406
    Sybase 407
  error handling 281
  functionality for JDBC objects 415
  loading 187
  overview 180
  performance 439
  registering with JDBC driver Manager 187
  specification supported 180
  specifying application IDs 279
  specifying connection URLs 188
  supported JDBC connection properties 201
  threading for driver 274
  tracking calls 181
  using connection pooling 192
  using data sources 200
  using Distributed Transaction
    Management 195
  using Spy with data sources 266
JNDI
  registering a SequeLink for JDBC
    DataSource object 466
  using for naming databases 191
JNDI 1.2 186
JTA 1.0.1 186

K

keyset-driven scrollable cursors
  (ODBC) 72, 73

L

LCASE function 367
LDAP
  configuring the ADO Client 114
  configuring the ODBC Client 33, 60
  creating a JDBC data source 191
  property in ADO 133
  setting parameters in DataDirect Test 212

specifying port for the listener 55, 114, 132
TCP/IP address of the LDAP server 54
UseLDAP attribute (ADO) 133
UseLDAP attribute (ODBC) 56
LDAP JNDI Provider 187
LEFT function 367
LENGTH function 367
libraries and header files, platform-specific 64
license file, defining path for .NET data provider 306
LicensePath 306
limiting the rows returned by a Select statement 486
Linux. *See* UNIX
Literal Bookmarks 168
literal parameter values 309
Literal Row Identity 155, 168
loading the JDBC driver 187
local transactions 311
Locale Identifier 161
LOCATE function 367
Lock Mode 155, 168
log example, Spy 267
LOG function 369
LOG10 function 369
logging JDBC calls 263
LogonID 54
long data, performance issues with retrieving 90
LTRIM function 367

M

managed code
use by applications that do not perform distributed transaction processing 312
ManagedConnectionFactory 273
managing
connections to improve driver performance 97
JDBC data sources 191
retrieval of database meta-information 88
mapping data types
ADO provider 172
SequeLink DbTypes 324
supported by DB2 on OS/390 378
supported by DB2 UDB 382
supported by Informix 9 389
supported by Microsoft SQL Server 393
supported by Oracle8 401
supported by Oracle9i 401, 406
supported by Sybase 407
master data source file
creating a 120
overview 118
Maximum Index Size 161
Maximum Open Chapters 161
Maximum Open Rows 155, 168
Maximum Pending Rows 155, 168
Maximum Row Size 161
Maximum Row Size Includes BLOB 161
Maximum Rows 155, 168
Maximum Tables in SELECT 161
Memory Usage 155, 168
meta-information
limiting amount to be retrieved 89
managing retrieval of 88
methods
.NET provider-specific 337
ADO Command object 154
ADO Recordset object 165
supported by ADO Connection object 157
supported public 326
tracing calls 313
Microsoft SQL Server
data types supported 393
support for scrollable cursors (ODBC) 72
MINUTE function 372
MOD function 369
Mode property 161
modifying ADO client data sources 115

MONTH function 372
MONTHNAME function 372
Multiple Connections property 161
Multi-Table Update 161
multithreaded applications
  cancelling functions 71, 274
  functionality of the ODBC driver 70

N

namespace, .NET data provider 319
naming databases with JNDI 191
native managed providers, performance advantages 353
NewPassword 54
non-pooled data source 466
NOW function 372
null arguments, impact on performance 85
NULL Collation Order 162
NULL Concatenation Behavior 162
Number
  raising to power 369
  rounding 369
numeric functions 368

O

object (.NET)
  exposing 308
  provider-specific prefix 326
  SequeLinkCommand 334
  SequeLinkCommandBuilder 336
  SequeLinkConnection 337
  SequeLinkError 339
  SequeLinkErrorCollection 340
  SequeLinkException 340
  SequeLinkInfoMessageEventsArgs 341
  SequeLinkTrace 343
  SequeLinkTransaction 311
objects supported
  by the .NET data provider 326
  by the JDBC driver 415
  by the SequeLink for ADO data provider 136
Objects Transacted 156, 168
ODBC
  functions that improve driver performance 94
  keyset-driven scrollable cursors 73
  Level 2.x API functions 65
  Level 3.x API functions 67
  optimizing performance 84
  static scrollable cursors 72, 73
ODBC Administrator
  configuring file client data sources 34
  configuring User and System client data sources 31
  starting 30
ODBC driver
  application IDs 78
  cancelling functions in multithreaded applications 71, 274
  configuring file client data sources 34
  connecting to a data source using a logon dialog box 39
  connection attributes 50
  connection options 29
  error handling 82
  handling Unicode characters 55
  ODBC API functions supported 65
  overview 29
  persisting result set as XML data file 80
  required libraries and header files 64
  specification supported 29
  threading 69
  using the system information file 46
ODBC translators 33, 60
odbc.ini (UNIX)
  example 46
  sample odbc.ini file 46
  using a centralized odbc.ini file 48
ODBCINI environment variable (UNIX) 47
ODBCTest 76

OLE DB
mapping methods to ADO 154
objects supported by the SequeLink for ADO provider 136
property groups supported 139
schema rowsets supported 138
OLE DB Services 162
OLE DB Version 162
Open Rowset Support 162
Oracle8
data types 401
support for scrollable cursors (JDBC) 277
support for scrollable cursors (ODBC) 72
using stored procedures with (ODBC) 74
Oracle9i
data types 401, 406
support for scrollable cursors (JDBC) 277
support for scrollable cursors (ODBC) 72
using stored procedures with (ODBC) 74
ORDER BY Columns In Select List 162
Others' Changes Visible 156, 168
Others' Inserts Visible 156
Output Parameter Availability 162
Own Changes Visible 156, 168
Own Inserts Visible 156, 168

P

parameter (.NET)
array support 309
in stored procedures 310
markers in stored procedures 351
markers, using the ? symbol 309
object
adding to a collection 309
properties supported 329
types 321
ParameterCollection object
parameter markers 309
properties and methods 329
ParameterMetaData object 442
Pass By Ref Accessors 162
Password
ODBC connection attribute 54
property for Connection object 162
performance hints
.NET data provider
avoiding use of CommandBuilder objects 349
managing connections 352
size of data retrieved 349
using a SequeLinkCommandBuilder object 336
using disconnected DataSet 353
using native managed providers 353
choosing the data type 350, 377
committing data 98
JDBC
fetching BigDecimal objects 284
penalty for emulating multiple open ResultSet objects 439
reducing download time 283
ODBC driver
avoiding the cursor library 96
catalog functions 84
locking a row when isolation level is Read committed 51
managing connections 97
managing retrieval of database meta-information 88
null arguments 85
reducing the size of retrieved data 90
retrieving data with SQLExtendedFetch 93
updating data in databases 99
using a dummy query 87
resetting the state of a connection 306
using bound columns 92
performance optimization
designing JDBC applications 293
retrieving data 288
selecting JDBC objects and methods 290
updating data 295
persisting
information 171
result set as XML data file (ODBC) 80

PI function 369
platform-specific header files and libraries 64
pool manager. *See* connection pooling
pooled data source 466
PooledConnection object 443
pooling connections 302
Port (ODBC connection attribute) 55
Port attribute 132
positional updates and deletes 99
POWER function 369
precedence of JDBC connection
  properties 200
prefix on a public object 326
Prepare Abort Behavior 162
Prepare Commit Behavior 162
PreparedStatement object 444
Preserve on Abort 156, 169
Preserve on Commit 156, 169
private assembly 308
Procedure Term 162
Prompt property for Connection object 162
properties
  ADO Command object 154
  ADO Connection object 158
  ADO Recordset object 167
  Data Source Information property
    group 140
  Data Source property group 140
  displaying ADO provider data source 109
  Initialization property group 145
  JDBC connection 200, 201
  Oracle data provider 333
  Session property group 152
  supported public 326
Property 147
property groups supported 139
Provider Friendly Name 162
Provider Name 163
provider string 128
Provider Version 163
Proxy Server, SequeLink 180
pseudo-columns 100

Q

QUARTER function 372
Quick Restart 156, 169
Quoted Identifier Sensitivity 163

R

RADIANS function 369
RAND function 369
Read Only Data Source 163
read-only result set 277
Recordset object
  dynamic properties 167
  methods 165
  overview 165
Ref object 449
Referenceable object 449
registering the JDBC driver 187
Remove Deleted Rows 156, 169
removing connections from a connection
  pool 304
renaming an ADO Client data source 115
REPEAT function 368
REPLACE function 368
Report Multiple Changes 156, 169
resetting connection state 306
resource adapters, JCA
  overview 196
  resource adapter class 273
  support 183
Resource Archive 185
result sets (JDBC)
  concurrency types 277
  types 276
  updatable supported 278
ResultSet object 449
ResultSetMetaData object 458

retrieving
data
avoiding long data 90, 348
reducing the size 90
using bound columns 92
using SQLExtendedFetch 93
database meta-information 88
long data using JDBC applications 288
warning information from .NET application 490
Return Pending Inserts 156, 169
reusing connections with connection pooling 192
RIGHT function 368
ROUND function 369
Row Privileges 156, 169
Row Threading Model 156, 169
rowset
characteristics 147
properties 147
using 171
Rowset Conversions on Command 163
RowSet object 459
RTRIM function 368

S

sample code
calling a stored procedure 489
limiting the rows returned by a Select statement 486
retrieving a result set using a DataAdapter object 484
retrieving a scalar value 491
retrieving warning information 490
updating data in a DataSet 487
using a distributed transaction 481
using a local transaction with a DataReader 479
using the CommandBuilder 483
SavePoint object (JDBC) 459
savepoint support in .NET data provider 344
saving current connection information 171
scalar functions
.NET code example of retrieving scalar values 491
support for 359
schema rowsets supported 138
Schema Term 163
Schema Usage 163
Scroll Backward 156, 169
scrollable cursors
concurrency types (JDBC) 277
insensitive (JDBC) 209
result set types (JDBC) 276
using keyset-driven (ODBC) 73
using static (ODBC) 73
using with JDBC driver 276
using with ODBC driver 72
scrolling backward 148
scroll-insensitive result sets (JDBC) 276
scroll-sensitive result sets (JDBC) 276
SECOND function 372
security (.NET)
attributes 345
enabling for tracing 313
required permissions for data provider 345
SequeLink Trace object 343
SequeLink documentation 20
SequeLink for .NET Client
data types
DB2 on OS/390 381
DB2 UDB 387
Informix 392
Microsoft SQL Server 399
Oracle 406
event handling 325
Framework data types 320
limiting number of rows returned 335
mapping parameter types 321
namespace 313, 319
public objects supported 326
SequeLinkCommand object 334
SequeLinkConnection object 319
thread support 325
using a distributed transaction 481

SequeLink for ADO provider
ADO Command object 154
connecting to a data source 122
connection attributes 130
data shaping 170
data source information properties 140
data source property group 140
data types
DB2 on OS/390 378
DB2 UDB 382
Informix 9 389
Microsoft SQL Server 393
Oracle8 401
Oracle9i 401, 406
Sybase 407
error handling 174
initialization properties 145
objects and interfaces 136
overview 105
rowset properties 147
rowset property group 147
session property group 152
SQL grammar supported 164
SequeLink for JDBC driver
JDBC 3.0 functionality 272
Proxy Server 180
specifying application IDs 279
SequeLink for ODBC driver
configuring file client data sources 34
connection dialogs 39
data types
DB2 on OS/390 378
DB2 UDB 382
Informix 9 389
Microsoft SQL Server 393
Oracle8 401
Oracle9i 401, 406
Sybase 407
functions that improve performance 94
isolation levels 413
overview 29
SequeLink Proxy Server
files 185, 186
overview 180
using SSL encryption 180
SequeLink Server errors (.NET) 346
SequeLink Server for ODBC Socket
JTA support 195
scrollable cursor support (JDBC) 277
support for scrollable cursors (ODBC) 72
SequeLinkCommand object 334
SequeLinkCommandBuilder object 336, 349
SequeLinkConnection object 337
SequeLinkError object 339
SequeLinkErrorCollection 340
SequeLinkException object 340
SequeLinkInfoMessageEventArgs object 341
SequeLinkTransaction object 311, 344
Serializable object 460
Server Cursor 156, 169
server errors, SequeLink (.NET) 346
Server Name 164
ServerDatasource 307
serviced components 354
session property group 152
sessions
maximum number supported 141
setting maximum number supported 158
setting
maximum number of sessions 158
ODBCINI environment variable (UNIX) 47
shared assembly 308
shell script, using to set environmental variables 47
SIGN function 370
SIN function 370
Skip Deleted Bookmarks 169
SLKStaticCursorLongColBuffLen 55
SOUNDEX function 368
SPACE function 368
specification supported
JDBC 180, 272
ODBC driver 29
OLE DB 144
specifying connection attributes 305

specifying provider-specific logon
information 160
Spy
about 263
attributes 265
log example 267
options 265
registering the JDBC driver 264
URL examples 267
URL syntax 265
using with JDBC data sources 266
SQL Server, support for scrollable cursors (ODBC) 72
SQL Support 164
SQLCancel, effect of threading 71, 274
SQLColumns, performance implications 87
SQLExecDirect 94
SQLExtendedFetch 93
SQLParamOptions 67, 95
SQLPrepare 94
SQLSetConnectAttr 79
SQLSpecialColumns 100
SQRT function 370
SSL encryption
cancel functionality 275
connection URL format 188
JAR files 184
permissions required 209
using with the proxy server 180
starting
Configuration Manager 109
ODBC Administrator 30
statement handles 98
Statement object 460
static scrollable cursors (ODBC) 72, 73
Status bar (Configuration Manager) 109
stored procedures
.NET code example 489
output array support 309
support for (.NET) 310
using parameter markers as arguments (.NET) 351
using the "?" in SQL statements (.NET) 309
using with ODBC driver 94
using with Oracle 74
string
changing case of 367
functions 366
length of 367
removing blanks from 367, 368
returning substring of 368
strong name 308
Strong Row Identity 156, 169
Struct object 462
Structured Storage 164
Subquery Support 164
SUBSTRING function 368
SupportLink 24
Sybase
.NET code example 478, 484, 489
data types 407
support for scrollable cursors (ODBC) 72
syntax for URLs for Spy 265
system information file
centralized 48
using to configure the ODBC driver 46

T

table characteristics, determining 87
Table Term 164
TAN function 370
TCP port 189
TCP/IP port for SequeLink listener 55
Technical Support, contacting 24
template data source file
creating a 119
overview 118
terminating the pool manager 474
testing
ADO connections 121
JDBC connections 215
ODBC connections on Windows 45

threading
.NET provider 325
ADO provider 142, 159
JDBC driver 274
ODBC driver use 69
time functions 370
TIMESTAMPADD function 372
TIMESTAMPDIFF function 372
tracing
.NET data provider method calls 343
overview 313
tracking JDBC calls 181, 266
Transaction DDL 165
transaction isolation levels 161
Transaction object
methods supported 329
properties supported 329
transactions
support for local 311
Translate button 33, 60
transliteration 60
transliteration support
selecting a translator for the ODBC Client 33
TRUNCATE function 370

U

UCASE function 368
Unicode
DB2 UDB data types supported 382
JDBC driver 280
see also *SequeLink Administrator's Guide*
support for ADO data provider 172, 384
Unique Rows 157, 169
UNIX
compiler requirements 65
configuring an ODBC data source 46
multithreading functionality of the ODBC driver 70
setting environmental variables 47
system information file 46
using a centralized odbc.ini or odbc64.ini file 48
Updatability 157, 169
updatable result sets 277
updates, positional 99
updating opened rowset 147
URL
examples (Spy) 267
format for connections (JDBC) 188
Use Bookmarks 157, 169
Use LDAP attribute (ADO) 133
UseLDAP attribute (ODBC) 56
USER function 373
User Name property of Connection object 165
using
centralized odbc.ini files (UNIX) 48
JDBC Client on Java 2 Platform 208
keyset-driven cursors 72
scrollable cursors
with the JDBC driver 278
with the ODBC driver 72
SequeLink for ADO Client 105
static cursors 72
stored procedures with Oracle 74

W

warning information, retrieving 490
WEEK function 372
window handle 146
Window Handle (ADO Connection object property) 165
Windows
configuring the SequeLink for ADO Client 105
multithreading functional of the ODBC driver 70
required ODBC libraries and header files 64
testing ODBC connections 45
WorkArounds attribute 56

X

XAConnection object 463
XADataSource object 463
XML, persistence, implementing (ODBC driver) 80

Y

year format for conversions 172
YEAR function 372

Z

z/OS. *See* OS/390